Génération de phrases multilingues par apprentissage automatique de modèles de phrases

Éric Charton

ACADÉMIE D'AIX-MARSEILLE
UNIVERSITÉ D'AVIGNON ET DES PAYS DE VAUCLUSE

THÈSE

présentée à l'Université d'Avignon et des Pays de Vaucluse
pour obtenir le grade de Docteur

SPÉCIALITÉ : Sciences Informatiques

École Doctorale I2S « Information Structure Systèmes »
Laboratoire d'Informatique (EA 931)

Génération de phrases multilingues par apprentissage automatique de modèles de phrases

par
Eric Charton

Soutenue publiquement à l'Université d'Avignon le 12 Novembre 2010 devant un jury composé de :

M^{me} Laurence Danlos	Professeur, Université Paris 7 (ALPAGE), Paris	Rapporteur
M. Guy Lapalme	Professeur, Université de Montréal (RALI), Montréal	Rapporteur
M. Laurent Besacier	Professeur, Université Joseph Fourier (IMAG), Grenoble	Examinateur
M. Jean-françois Bonastre	Professeur, Université d'Avignon (LIA), Avignon	Examinateur
M. Michel Gagnon	Professeur, École Polytechnique de Montréal (GIGL), Montréal	Examinateur
M. Juan-Manuel Torres-Moreno	MdC (HDR), Université d'Avignon et des Pays de Vaucluse (LIA), Avignon	Directeur

---

Laboratoire d'Informatique d'Avignon

Remerciements

Mes plus chaleureux remerciements vont en premier lieu à Jean-François Bonastre. Sans ses encouragements et sa bienveillance, lors de toutes les étapes de cette aventure, c'est certain, rien n'aurait été possible. Ils vont ensuite au Dr Juan-Manuel Torres-Moreno qui a bien voulu m'accompagner dans la démarche complexe et prenante que constitue ce travail de recherche.

Je suis particulièrement reconnaissant à tous ceux, membres ou non de la communauté académique qui m'ont aidé sans autre revendication que celle de me rendre service. Merci à Georges Linares de m'avoir fourni tout le matériel dont j'avais besoin pour mes expériences et d'avoir pris en charge certains aspects logistiques, à Patricia Velazquez-Morales d'avoir contribué à la mise au point de mes expériences et activement aidé pour la finition d'aspects importants de ce travail, à Frédéric Béchet pour sa curiosité active envers mes travaux. Un remerciement très spécial va au Dr Nimaan Abdillahi qui m'a invité à mettre un peu d'Afrique dans ce travail (c'était un rêve).

Je ne voudrais pas oublier de témoigner de ma sympathie envers tous les membres du Laboratoire Informatique d'Avignon et du CERI que j'ai côtoyé au cours de ces quelques années. Je n'en cite aucun pour n'en oublier aucun ; ils habitent maintenant un peu partout sur la planète dans des maisons et même dans des cubicles (en particulier à Singapour, San Fransisco, Brisbane, Mexico, Montréal, Djibouti, Sao Polo, mais aussi à Marseille, au sud de Marseille, à Toulouse, Aix en Provence, au Mans, à Chateaurenard, à Caumont, Orange et en Avignon, évidemment) certain(e)s sont devenus des ami(e)s et ils se reconnaîtront.

Et je n'oublie surtout pas la patience et l'affection de Antoine, Charles et surtout Marie-Jean, éminents membres de ma tribu de gens du voyage désormais devenue cosmopolite !

Résumé

La Génération Automatique de Texte (GAT) est le champ de recherche de la linguistique informatique qui étudie la possibilité d'attribuer à une machine la faculté de produire du texte intelligible. Dans ce mémoire, nous présentons une proposition de système de GAT reposant exclusivement sur des méthodes statistiques. Son originalité est d'exploiter un corpus en tant que ressource de formation de phrases. Cette méthode offre plusieurs avantages : elle simplifie l'implémentation d'un système de GAT en plusieurs langues et améliore les capacités d'adaptations d'un système de génération à un domaine sémantique particulier. La production, d'après un corpus d'apprentissage, des modèles de phrases finement étiquetées requises par notre générateur de texte nous a conduit à mener des recherches approfondies dans le domaine de l'extraction d'information et de la classification. Nous décrivons le système d'étiquetage et de classification de contenus encyclopédique mis au point à cette fin. Dans les étapes finales du processus de génération, les modèles de phrases sont exploités par un module de génération de texte multilingue. Ce module exploite des algorithmes de recherche d'information pour extraire du modèle une phrase pré-existante, utilisable en tant que support sémantique et syntaxique de l'intention à communiquer. Plusieurs méthodes sont proposées pour générer une phrase, choisies en fonction de la complexité du contenu sémantique à exprimer. Nous présentons notamment parmi ces méthodes une proposition originale de génération de phrases complexes par agrégation de proto-phrases de type *Sujet,Verbe,Objet*. Nous envisageons dans nos conclusions que cette méthode particulière de génération puisse ouvrir des voies d'investigations prometteuses sur la nature du processus de formation de phrases.

Les travaux de recherche présentés dans ce mémoire ont été menés au Laboratoire Informatique d'Avignon jusqu'en octobre 2009. Ils ont été complétés et mis en application à l'École Polytechnique de Montréal à partir du 1er décembre 2009, dans le cadre des travaux du projet Gitan, dirigé par les Professeurs Michel Gagnon et Benoit Ozell. L'ensemble de cette thèse à été menée sous la direction scientifique du Dr Juan-Manuel Torres-Moreno (MdC HDR).

Mots clés

Génération Automatique de Texte, Génération de phrases, Apprentissage automatique, Syntaxe, Extraction d'information, Agrégation

Abstract

Multilingual Natural Language Generation using sentence models learned from corpora

Natural Language Generation (NLG) is the natural language processing task of generating natural language from a machine representation system. In this thesis report, we present an architecture of NLG system relying on statistical methods. The originality of our proposition is its ability to use a corpus as a learning resource for sentences production. This method offers several advantages : it simplifies the implementation and design of a multilingual NLG system, capable of sentence production of the same meaning in several languages. Our method also improves the adaptability of a NLG system to a particular semantic field. In our proposal, sentence generation is achieved trough the use of sentence models, obtained from a training corpus. Extracted sentences are abstracted by a labelling step obtained from various information extraction and text mining methods like named entity recognition, co-reference resolution, semantic labelling and part of speech tagging. The sentence generation process is achieved by a sentence realisation module. This module provide an adapted sentence model to fit a communicative intent, and then transform this model to generate a new sentence. Two methods are proposed to transform a sentence model into a generated sentence, according to the semantic content to express. In this document, we describe the complete labelling system applied to encyclopaedic content to obtain the sentence models. Then we present two models of sentence generation. The first generation model substitute the semantic content to an original sentence content. The second model is used to find numerous proto-sentences, structured as *Subject, Verb, Object*, able to fit by part a whole communicative intent, and then aggregate all the slected proto-sentences into a more complex one. Our experiments of sentence generation with various configurations of our system have shown that this new approach of NLG have an interesting potential.

This research have been mainly conducted in the Computer Laboratory of Avignon University, until October 2009. It was supplemented and applied at the École Polytechnique de Montreal starting from December 1st, 2009, within the framework of the Gitan project, directed by Professors Michel Gagnon and Benoit Ozell. The whole work described in this thesis has been conducted under the scientific supervision of Dr. Juan-Manuel Torres-Moreno (MdC HDR).

Keywords

Natural Language Generation, Sentence generation, Statistical learning, Syntax, Information extraction, Aggregation

Table des matières

Introduction

Toute production humaine, et celle du texte n'y échappe pas, est soumise au crible de l'évolution de nos sociétés qui veut que tout acte de production puisse être amélioré par une voie qui réduise son coût et augmente sa productivité. C'est donc tout naturellement que l'activité de mécanisation du calcul, qui s'est progressivement penchée sur la plupart des tâches de nature humaine, s'est rapidement intéressée à la potentialité d'une mécanisation du langage.

Historique de la Génération Automatique de Texte

Dès 1945, la génération de texte s'inscrit dans le cadre large du traitement automatique de la langue. La fin des années 40 voit naître un champ d'expérimentation scientifique nouveau, regroupé sous le vocable d'Intelligence Artificielle (IA) (Anis, 1994). On retiendra de cette période que les deux premières évocations de la question de la génération de texte par traitement informatique peuvent être attribuées successivement à Alan Turing et à Claude Shannon. L'un et l'autre mènent très tôt une réflexion sur le dialogue pour Turing et sur l'assemblage automatique de mots pour Shannon. Alan Turing réfléchit à cette possibilité dans le cadre large d'un système de dialogue. La proposition du test d'IA intitulé *Le jeu de l'imitation* que Turing décrit en 1950 dans *Computing machinery and intelligence* (Turing, 1950), consiste à faire dialoguer un humain avec un ordinateur et ce même humain avec un autre humain. Selon Turing, si l'homme qui engage les deux conversations n'est pas capable de différencier un locuteur humain d'un ordinateur, il est permis de considérer que le logiciel a passé avec succès le test. Si Turing inscrit son travail dans un cadre large (qui doit répondre à la question *«Les machines peuvent elles penser ?»*), son test va rapidement devenir une expérience codifiée de génération automatique dans un contexte de dialogue [1]. Avant lui, en 1948, Shannon avait posé les bases de la modélisation statistique du langage, en introduisant son article sur la *Théorie Mathématique de la Communication* (Shannon, 1948) par un long développement sur la GAT. Il proposait de construire arbitrairement des phrases d'apparence grammaticale correcte en utilisant des phénomènes combinatoires [2].

1. le *Loebner Prize for artificial intelligence*, qui reprend le principe du test de Turing sous la forme d'une campagne d'évaluation de systèmes est intégré sous forme d'atelier de la conférence IEEE Interspeech en 2009.
2. Nous décrivons plus en détails cette expérience et ses résultats dans le chapitre 2.

Premières expériences de Génération dans le contexte de la traduction automatique

Ces travaux précurseurs s'inscrivent dans une perspective de compréhension des phénomènes de production linguistique avec les moyens de calculs de l'époque. C'est en tant que composant des systèmes de *Traduction Automatique* (TA) que la GAT va vivre une première phase de développement. Pendant la décennie 1955-1965, dans le contexte de la guerre froide, la TA devient la locomotive de la linguistique formelle et de ce qui va devenir la linguistique computationnelle. A la fin des années 50, la publication de *Structures Syntaxiques* (Chomsky, 1957) par Noam Chomsky offre de nouvelles perspectives de recherches dans le domaine de la GAT.

En 1961, Yngve est le premier à explorer les applications de GAT que les propositions de Chomsky semblent permettre. Il les décrit dans son article intitulé *Random Generation of English Sentences* (Yngve, 1961). Mais les phénomènes de complexité linguistique sont un obstacle : Yngve indique que les *»grammaires transformationnelles originelles ont été abandonnées car elles ne peuvent pas être mécanisées avec un appareil fini, en raison de la difficulté d'associer une structure de phrase au résultat d'une transformation»*. Les travaux de (Matthews, 1962), similaires à ceux de Yngve, s'inscrivent encore pour le moment dans la perspective d'insérer un composant de GAT dans un système de traduction. Le premier véritable travail de génération dans une optique théorique et non appliquée à la TA pourrait être attribué à Friedman, qui publie en 1969 son article intitulé *Direct random generation of sentences* (Friedman, 1969) . Le système proposé, écrit en Fortran et fonctionnant sur IBM 360/67 est un générateur aléatoire de phrases dont les fondements théoriques revendiqués sont les modèles linguistiques rénovés et enrichis proposés par Chomsky dans *Aspects de la théorie syntaxique* (Chomsky, 1969). Le projet est de créer un composant de production de texte susceptible de mettre en application les grammaires transformationnelles. Le générateur de phrases utilise une définition sommaire d'un arbre représentant l'*Intention de Communication*, le programme se chargeant ensuite de remplir l'arbre. Le programme décrit se contente de produire des structures syntaxiques et n'est pas testé dans un contexte d'évaluation de phrases sémantiquement intelligibles. Le principe qu'il met en œuvre demeure d'actualité et est implémenté sous des formes plus ou moins sophistiquées dans nombres de systèmes de GAT modernes.

De la production de grammaire à la production de phrases

Ces premiers systèmes de GAT sont généralement considérés par la littérature comme des «*systèmes sous influence de la linguistique chomskyenne*» [3]. Leur finalité est essentiellement de tester les grammaires génératives et ils considèrent peu ou pas la problématique de la production de phrases sémantiquement correctes et intelligibles, d'après une information précise à communiquer. On en voudra pour preuve que jusqu'au milieu des années 70, les systèmes de GAT expérimentaux remplacent souvent la phase

3. Voir notamment (Sabah et Zock, 1992) p10.

de structuration de l'*Intention de Communication* par un processus aléatoire (Friedman, 1969; Matthews, 1962; Yngve, 1961). Mais la publication du rapport ALPAC en 1966 [4] marque l'arrêt du financement de la traduction automatique jugée peu prometteuse et donc de toutes les recherches sur les éléments qui entrent en jeu dans la TA, telle la GAT.

C'est alors une nouvelle génération de systèmes, plus proches de la recherche théorique sur la production langagière, qui va émerger vers la fin des années 70 et de la décennie 80. Faisant écho à la distinction faite entre le «que dire» (l'intention à communiquer) et le «comment le dire» (la phrase ou le groupe de phrases qui traduisent cette intention) formalisée par (Thompson, 1977), les auteurs proposent des systèmes à deux familles de composants de production principaux, décrits dans la littérature comme *composants stratégiques et tactiques* [5]. Les composants stratégiques sont chargés de gérer l'information brute et les composants tactiques sont dédiés à la transformation de cette information en un texte intelligible. Il est ici question de préparer et structurer des données dans un premier composant («que dire»), puis de les transmettre à un second composant linguistique («comment le dire») qui se charge de les transformer en phrases. L'architecture est encore rudimentaire : elle ne gère pas tous les aspects de la complexité des phrases, de leur variété lexicale ou des procédés rhétoriques qu'elles peuvent mettre en oeuvre. Elle préfigure cependant ce que seront les architectures des systèmes futurs.

À partir des années 90 ces architectures prennent en charge des aspects toujours plus fins de la GAT en exploitant la puissance croissante des systèmes d'informations. Progressivement, il devient de plus en plus difficile de faire la distinction entre *composants tactiques* et *composants stratégiques*. La tâche de GAT étant particulièrement complexe et ses fondements théoriques encore peu stables, on observe au fil de l'évolution des systèmes de multiples déplacements dans les rôles attribués à ces composants. Ce phénomène touche particulièrement les *composants stratégiques* qui se voient au gré des publications attribuer des fonctions de plus en plus complexes comme par exemple l'agrégation de plusieurs phrases en une seule. Les *composants tactiques*, qui sont plus proches de la syntaxe, ont conservé une relative stabilité dans leur définition. Cette stabilité aura aussi pour conséquence de ralentir l'émergence de solutions neuves prenant en compte notamment les avancées obtenues en matière de Traitement Automatique de la Langue (TAL) statistique.

Quoi qu'il en soit, le saut technologique de ces systèmes vient de ce qu'ils ne se satisfont plus de démontrer leur aptitude à générer des phrases syntaxiquement correctes. Ils explorent désormais les facultés langagières de la GAT dans une perspective applicative et productiviste. Ces générateurs, encore rudimentaires, cherchent à démontrer la capacité du système d'information à automatiser la production de texte, au même titre qu'il a déjà automatisé la manipulation de chiffres ou la production industrielle.

4. *Languages and machines : computers in translation and linguistics.* A report by the Automatic Language Processing Advisory Committee, National Research Council, 1966. (Publication 1416.).
5. Lire notamment (Reiter et Dale, 2000) page 115 et (Shaw, 2002b) page 16

Résumé : sur l'histoire de la GAT

L'histoire de la GAT vue sous l'angle de l'épistémologie nous montre que la question de la production automatisée de texte est étudiée dès que naît la recherche en linguistique computationnelle, dans une perspective de compréhension des phénomènes calculatoires liés au langage. Comme la plupart des autres activités du domaine du TAL, la GAT est concernée dès les années 50 par la divergence entre l'approche de la *Théorie Mathématique de la Communication* de Shannon et l'approche chomskyenne de la théorie syntaxique. Nous avons pu constater que la GAT avait connue très tôt une période d'épanouissement, dans les années 50 et 60, lorsqu'elle était considérée comme un composant d'un système de traduction. Les chercheurs n'ont alors pas hésité à déployer des méthodes hybrides de génération de texte impliquant à la fois des grammaires génératives et des systèmes statistiques. À cette période de recherche importante a succédé - à partir des années 70 - un relatif désintérêt pour la linguistique computationnelle, conséquence du tarissement des financements militaires et institutionnels. Ce ralentissement fût suivi d'une lente reprise dès les années 90. Nos travaux s'inscrivent dans cette nouvelle période de développement de l'activité scientifique autour de la GAT qui voit émerger des communautés structurées et actives autour de groupes d'intérêt dynamiques (tel le SIGGEN [6] par exemple, qui organise une conférence spécialisée, INLG, parrainée par ACL).

Contributions

Notre principale contribution est l'intégration au sein d'une architecture de GAT d'un composant de génération utilisant des phrases pré-existantes extraites d'un corpus et rendues les plus abstraites possibles par des méthodes d'étiquetage. Nous avons postulé qu'il existe dans de grands corpus une vaste quantité de phrases ou d'éléments de phrases déjà construits qui sont susceptibles d'être réemployés et transformés pour s'adapter à une *Intention de Communication*. Cette proposition offre l'avantage de remplacer la phase d'écriture de règles de génération d'un système de GAT par un apprentissage automatique, ce qui simplifie son processus de conception et d'adaptation. Elle permet notamment d'envisager que l'élaboration de systèmes de GAT multilingues et leur adaptation à un domaine sémantique, puissent être confiées à un processus largement automatisé. Nos travaux nous ont conduits à mettre au point un ensemble de ressources exploitables dans un système de GAT, diffusées de manière ouverte. Ces ressources sont constituées de corpus et de classes de génération développées en Java.

6. Consulter www.siggen.org.

Organisation de ce mémoire

Ce mémoire est divisé en trois parties. La partie I contient une étude de l'état de l'art actuel de la GAT et s'achève par une description du système que nous proposons. La seconde et la troisième partie consistent en une présentation des différents composants du système de GAT proposé.

Dans la partie I nous commençons par décrire dans le chapitre 1 les architectures existantes des systèmes de GAT, puis les composants stratégiques d'un générateur de texte. Nous décrivons ensuite, dans les chapitres 2 et 3, les deux approches principales de génération de phrases. La première approche utilise des méthodes recourant à des modules de génération implémentés sous forme de règles de production issues d'une théorie linguistique. La seconde approche exploite une méthode de nature statistique. Nous terminons cette partie en présentant dans le chapitre 4 notre système à architecture *pipeline* exploitant des ressources obtenues par apprentissage automatique d'après un corpus.

Nous décrivons dans la partie II les divers éléments d'extraction d'information utilisés pour obtenir d'après un corpus un ensemble de *phrases modèles* finement étiquetées et rendues les plus abstraites possibles afin de servir de support à un processus de génération de texte. Après avoir dressé dans le chapitre 5 un panorama des techniques de TAL disponibles pour modéliser des ressources linguistiques, nous présentons successivement dans le chapitres 6 une ressource lexicale et conceptuelle extraite d'après l'encyclopédie Wikipédia, dans le chapitre 7 un système d'étiquetage par entités nommées mis au point d'après cette ressource, et dans le chapitre 8 un extracteur de coréférences qui permet de compléter notre système.

Dans la partie III, nous présentons un générateur de texte qui utilise des modèles de phrases dans un composant de génération linguistique. Nous décrivons dans le chapitre 9 les algorithmes d'extraction d'information utilisés pour identifier dans un *Corpus de Phrases Modèles (CPM)* une structure de phrase pré-existante pouvant servir de support à l'expression d'une *Intention de Communication (IC)*. Nous présentons deux méthodes de génération automatique de texte. La première, décrite au chapitre 9, repose sur l'intégration d'une IC dans une phrase complète. La seconde, décrite au chapitre 10, s'appuie sur des assemblages de proto-phrases de type *Sujet, Verbe, Objet* (SVO) issues d'un corpus d'apprentissage pour générer une phrase unique par des méthodes d'agrégation.

Le chapitre 12 présente une synthèse de la problématique particulière de l'évaluation en GAT. Il s'achève par une évaluation de notre système. Nous concluons en décrivant les possibilités de développement de notre approche.

Première partie

Théorie et génération

Chapitre 1

Structures et méthodes des systèmes de Génération Automatique de Texte

Sommaire

La génération de texte consiste à transformer une ou plusieurs informations brutes en une forme particulière de support d'expression, la phrase. Fondamentalement la GAT inclut donc à deux extrémités d'une chaîne de traitement, d'un côté une intention de communication qui peut prendre la forme d'une information structurée ou d'une donnée, et à l'autre extrémité une phrase ou un groupe de phrases générées. Pour décrire cette notion d'intention de communication, (Sabah et Zock, 1992) utilisent cette formulation : «*comme la plupart des activités des systèmes vivants, le comportement verbal est motivé par des buts. La situation extérieure, c'est-à-dire l'état du monde et les états physiques, émotionnels et cognitifs des participants, induisent certains besoins qui demandent à être satisfaits*». L'idée d'*Intention de Communication* existe dans de nombreuses branches de la psycholinguistique ou du traitement de la langue ((Pinker et Jackendoff, 2005) la résument par l'expression «*représentation mentale sous la forme de structure conceptuelle*»). Elle a été reprise récemment par (Hauser et al., 2002; Fitch et al., 2005) et intégrée au *programme minimaliste* sous le terme de *Conceptual Intentional* (Chomsky, 2005). Pour (Jackendoff et Pinker, 2005) le rapport entre phrase et intention de communication peut être

décrit comme *une solution à un problème basique de conception* [1] *: «les relations sémantiques sont récursives et multidimensionnelles et doivent être représentées sous une forme linéaire dans une phrase»*. L'*Intention de Communication* est donc une abstraction qui peut prendre de multiples formes : des données numériques, des entrées de bases de données ou encore des groupes de mots reliés hiérarchiquement. Dans un système de génération appliqué à la météorologie, par exemple, il s'agira de données numériques (une température, une probabilité) ou alphabétiques (noms de nuages, d'un vent, type de temps prévu). Avec un système à base de règles, l'*Intention de Communication* peut être décrite par un sujet, un verbe et un complément (exemple : `Henri, Manger, Ananas`) des instructions de conjugaison (exemple : temps passé) et c'est le système de GAT qui prendra en charge la fabrication d'une phrases (`Henri a mangé un ananas`).

Dans les systèmes les plus récents, l'*Intention de Communication* peut inclure un concept décrit par une représentation ontologique (comme dans le système SimpleNLG (Reiter et Gatt, 2009) décrit plus loin dans le chapitre 2). L'application de GAT peut alors incorporer des processus susceptibles de paramétriser l'*Intention de Communication* selon des règles complexes : un générateur de biographies peut extraire de la description contenue dans l'instance ontologique de nombreuses informations (une date de naissance, des noms de membres de la famille, des éléments historiques), puis les intégrer et les mettre en forme au sein d'un assemblage complexe de phrases. Pour illustrer ce principe, considérons la description d'une objet sous une forme simplifiée (table 1.1), ici la description d'une habitation, que nous voudrions représenter avec un système de GAT.

New Maison
Maison.Propriétaire=John ; Maison.Attributs=Blanche ;Grande ; Maison.Situation=Rue de Paris ;proche de l'école primaire ;

TABLE *1.1* – *Exemple d'Intention de Communication.*

Il existe plusieurs possibilités de transformer cette représentation en phrases. La phase de *planification*, dans sa forme la plus rudimentaire, pourrait déjà produire :

```
John possède une maison. La maison de John est blanche. La maison
de John est grande. La maison de John est située rue de Paris.
La maison de John est proche de l'école primaire.
```

À un autre stade, la planification peut être complétée par une étape de *lexicalisation* dont le rôle sera d'enrichir le vocabulaire comme ceci :

```
John [possède;détient] une [maison;demeure]. La [maison;demeure]
de John est blanche. La [maison;demeure] de John est [grande;spacieuse].
```

Une phase de production d'expressions co-référentes peut également être introduite comme ci-dessous :

1. *«a basic design problem»* (Jackendoff et Pinker, 2005).

```
John [possède;détient] une [maison;demeure]. [Elle|La{maison;demeure}]
est blanche. [[Sa][maison;demeure]||[elle]] est [grande;spacieuse].
```

Finalement on pourra aussi intégrer toutes ces informations dans une seule phrase par un processus d'*agrégation* comme suit :

```
John possède une grande maison blanche située rue de Paris, à
proximité de l'école primaire.
```

On constate que toutes ces étapes font appel à des ressources et invoquent des processus variés : elles impliquent des besoins lexicaux, des méthodes de formation linguistique ou de calcul mathématique, des techniques de détection et de production d'expressions de substitution. Ceci illustre pourquoi la conception d'un système de GAT revient presque toujours :

1. à choisir un formalisme pour représenter l'*Intention de Communication (IC)* ;
2. à adopter une architecture en *pipeline*, c'est-à-dire une segmentation des tâches en *composants* successifs et une définition des échanges d'informations entre ces *composants* ;
3. à choisir les théories que les *composants* de cette architecture implémentent pour produire finalement une phrase syntaxiquement et sémantiquement correcte.

1.1 Formalisation de l'*Intention de Communication*

On ne voit pas émerger dans la littérature, comme souvent en GAT, de formalisme bien défini pour décrire une IC. Le module de réalisation de surface SURGE (Elhadad et Robin, 1996), sans répondre à aucune norme formelle, propose un système de type squelette, partiellement lexicalisé par des arbres thématiques spécifiant les rôles sémantiques, avec des éléments lexicaux ouverts et une catégorie syntaxique associée à chaque constituant. Ceci nous donne l'exemple de la figure 1.1. C'est cette structure, très proche de la syntaxe mais ne répondant pour le moment à aucune norme précise, qui est ensuite transmise au module de réalisation de phrase SURGE.

Au début des années 2000, (Shaw, 2002b)[2] remarque que les chercheurs utilisent pour formaliser les *représentations sémantiques*[3] destinées à la GAT aussi bien les grammaires SFG, que TAG, ou encore la Théorie Sens-texte, cherchant ainsi à s'appuyer sur des théories linguistiques solides. Néanmoins Shaw observe qu'aucune étude comparative sérieuse n'a été réalisée jusqu'ici (nous n'avons pas trouvé trace d'un tel travail qui serait intervenu postérieurement) sur l'adéquation de ces formalismes avec la tâche spécifique de génération de texte. Dans le system MAGIC, par exemple le *Planificateur de phrase* CASPER (*Clause Aggregation in Sentence PlannER*) exploite une représentation symbolique de l'IC (Shaw, 2002b) qui repose sur une structure, à base de prédicats, influencée par les grammaires LFG (voir la figure 1.2).

$$\begin{bmatrix} cat = clause & \\ process & \begin{bmatrix} type = composite \\ relation = possessive \\ lex = |hand| \end{bmatrix} \\ partic & \begin{bmatrix} agent & \begin{bmatrix} cat = pers-pro \\ gender = feminine \end{bmatrix} \\ affected\,[1] & \begin{bmatrix} cat = pers-pro \\ gender = feminine \end{bmatrix} \\ possessor\,[1] & \\ possessed & \begin{bmatrix} cat = np \\ lex = |draft| \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

FIGURE 1.1 – *Représentation formelle de la phrase* ***She hands the draft to the editor*** *selon (Elhadad et Robin, 1996).*

$$\begin{bmatrix} pred & \begin{bmatrix} type = event \\ concept = manger \\ temse = present \end{bmatrix} \\ arg1 & \begin{bmatrix} type = entity \\ concept = personne \\ name = Jean \end{bmatrix} \\ arg2 & \begin{bmatrix} type = attribute \\ concept = satisfied \end{bmatrix} \end{bmatrix}$$

FIGURE 1.2 – *Représentation formelle de la phrase* ***Jean mange*** *selon (Shaw, 2002b).*

Finalement, notre examen du corpus scientifique sur ce sujet nous pousse à conclure que la plupart des propositions adoptent pour représenter une IC un formalisme très proche des Grammaires Lexicales Fonctionnelles (*Lexical-Functional Grammars LFG*) de (Kaplan et Bresnan, 1987). Shaw remarque d'ailleurs en comparant le formalisme SFG avec celui des LFG que le formalisme à prédicats des LFG est plus aisé à manipuler. Il souligne que les structures du LFG permettent notamment de mettre en œuvre les structures sémantiques décrites dans (Jackendoff, 1992). Le squelette de LFG repose sur un formalisme grammatical à structures fonctionnelles dont le squelette est celui des grammaires non contextuelles. La proposition dans les années 80 de ce modèle par Kaplan et Bresnan était justifiée par un besoin de dépasser certains points faibles des grammaires génératives et transformationnelles, notamment leur complexité excessive et leur difficulté d'application à des langues structurées plus librement que l'anglais ou les langues Romanes. On retrouve une forme dérivée de ce formalisme dans (Reiter et Dale, 2000)[4] mais libellée sous la forme *Sujet, objet* ou encore *Sujet, prédicat* selon les situations, plus proche des formes logiques du premier ordre décrites sous le nom de *f-structure* par (Kaplan et Bresnan, 1987) (voir la figure 1.3).

2. Voir notamment le chapitre 2.2, p24 et suivantes.
3. NdEC : Shaw utilise l'expression *meaning representation*.
4. Lire notamment page 120, architecture du composant de Microplanning, ainsi que la page 69, Abstract Syntax.

```
// Jean eats
SPhraseSpec p = new SPhraseSpec();
p.addSubject("Jean");
p.setVerb("eat");
p.setTense(Tense.PRESENT);
Realiser r = new Realiser();
String output = r.realiseDocument(p);
System.out.println(output);
```

TABLE 1.2 – *Programmation de l'Intention de Communication* ***Jean eats*** *dans SimpleNLG.*

$$\begin{bmatrix} Type = PSAbstractSyntax \\ Head = be \\ features = \begin{bmatrix} tense = past \end{bmatrix} \\ Subject = \begin{bmatrix} type = PSAbstractSyntax \\ head = |month| \\ feature = [definite : true] \end{bmatrix} \\ predicate = \begin{bmatrix} type = PSAbstractSyntax \\ head = |warm| \\ features = [Comparative : +] \\ modifiers = |slightly| \\ object = |average| \end{bmatrix} \end{bmatrix}$$

FIGURE 1.3 – *Représentation formelle de la phrase* ***The month was slightly warmer than average****. Exemple issu de (Reiter et Dale, 2000).*

1.2 L'architecture *pipeline* des systèmes de génération

Dans l'organisation modulaire des systèmes de GAT en mode *pipeline*, plusieurs composants traitent chacun séquentiellement une partie du problème de la génération en partant de la représentation la plus abstraite des IC jusqu'à aboutir à une forme écrite intelligible, syntaxiquement et sémantiquement correcte. Au milieu des années 90, (Reiter, 1994) propose une synthèse de la littérature sur le problème de l'architecture idéale. Pour cet auteur les composants d'une architecture modulaire qui feraient consensus incluent la prise en charge des étapes suivantes :

- Définition du contenu (*Content Planning*) : ce module permet de faire correspondre l'entrée initiale du système de GAT, l'IC, avec une représentation sémantique (par exemple les éléments conceptuels de base d'une réponse à une question), le cas échéant complétée par des annotations rhétoriques (Reiter précise que ces annotations peuvent correspondre à la *Rhetorical Structure Theory (RST)*).
- Planification des phrases (*Sentence Planning*) : ce module est connu sous plusieurs noms dans la littérature (Reiter retient celui proposé par (Rambow et Korelsky, 1992) pour le système JOYCE). La fonction de ce module est de faire correspondre la représentation sémantique avec une forme linguistique. Ceci inclut la gestion des anaphores, le choix lexical, la définition des relations grammaticales et la

structuration des informations en syntagmes et relations de dépendances.
- Génération de surface (*Surface Generation*) : ce module prend en entrée une représentation de l'IC (incluant les éléments collectés par le module de définition de contenu et le planificateur de phrases) et produit une forme de surface qui communique cette information. Il y a dans ce module, tel qu'il est présenté par Reiter, une apparente redondance entre le planificateur de phrase et le générateur de surface. Reiter justifie cette séparation par la présence de planificateur et de générateur de phrases dans «*tous les systèmes qu'il a étudié*» [5].
- Génération structurelle (*Structure Realisation*) [6] : elle concerne la morphologie et le formatage. Reiter observe que la plupart des systèmes possèdent un composant morphologique relativement simple. Le module de formatage (présent dans les systèmes IDAS, JOYCE et Penman) est exclusivement dédié à la présentation finale, c'est-à-dire à la mise en forme pour des applications (LaTeX, sorties HTML, etc.). Le formatage relève, selon nous, essentiellement du travail d'ingénierie.

Finalement, de son questionnement sur l'architecture la plus adaptée à la GAT, Reiter ne retient que deux composants de génération véritablement indispensables : celui dédié à la structuration de l'IC et un composant de transformation de l'IC en *forme de surface* d'une phrase.

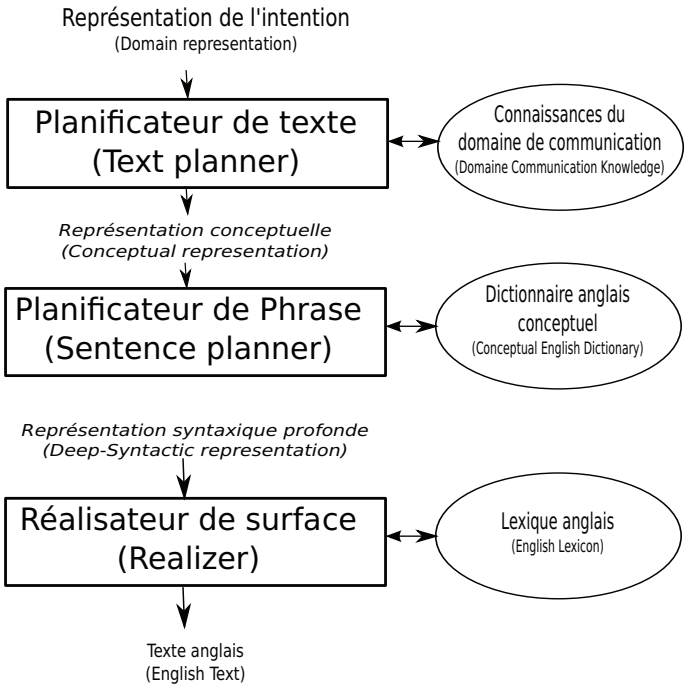

FIGURE *1.4* – *L'architecture de JOYCE proposée dans (Rambow et Korelsky, 1992) (avec entre parenthèses les libellés anglais de l'illustration originale).*

5. Dans (Reiter, 1994), «*All of the examined system had separate sentence-planning and surface generation modules, and the various intermediate forms are used to pass between these module similar kinds of information*».
6. Ce terme apparaît dans (Reiter et Dale, 2000) à la page 59 du chapitre sur l'architecture.

On retrouvera cette architecture en *pipeline* simplifiée dans les systèmes de génération à composants statistiques (voir par exemple (Knight et Hatzivassiloglou, 1995)). Le système JOYCE (Rambow et Korelsky, 1992), pour sa part, adopte une structure à trois modules légèrement adaptée. Elle est composée d'un *Plannificateur de texte* qui produit une représentation conceptuelle, transmise à un *Plannificateur de phrase* qui produit une représentation structurée profonde, elle même transmise à un *Générateur de surface* qui produit une phrase intelligible (figure 1.4). Le succès de cette proposition conduit Reiter et Dale à proposer une architecture (voir en figure 1.5) qui en dérive (Reiter et Dale, 1997) et qui est aujourd'hui régulièrement citée en tant que référence dans la littérature diffusée postérieurement aux années 90[7]. Trois composants principaux y prennent place, agencés en *pipeline* : un *Planificateur de texte*, un *Planificateur de phrases* et un *Générateur de formes de surfaces*. En réalité, dans les systèmes les plus récents, l'intrication du *Planificateur de texte* (qui gère toutes les phrases d'un document) et du *Planificateur de phrases* laisse place à un composant de transformation d'une *Intention de Communication* lexicalisée en forme de surface d'une phrase, comme dans SimpleNLG (Reiter et Gatt, 2009). Aujourd'hui encore et malgré l'esquisse de consensus décrite, l'ordre et l'affectation des tâches dévolues à chaque module d'un générateur de texte à architecture *pipeline* demeurent peu standardisés. Il est courant de voir des systèmes qui fusionnent plusieurs tâches dans un seul composant et donc souvent un seul processus. Ceci est particulièrement vrai dans le cadre général de ce que nous intitulons la planification de l'*Intention de Communication*. Ainsi, comme le rappellent (Danlos et Roussarie, 2000)[8] *«dans la littérature, on peut retrouver les tâches de Sélection du Contenu Profond et de Structuration Rhétorique réunies sous le chef de macro-planificateur, planificateur de texte, générateur profond ; de même le module de Planification de Phrase [...], encore nommé micro-planificateur ou module linguistique et dont l'objet est de mettre en place le matériau linguistique qui va «incarner» le message, contient communément les tâches de Planification Syntaxique et de Lexicalisation»*.

1.3 Composants stratégiques de production de phrase de l'architecture pipeline

Les architectures *pipeline* qu'elles soient simplifiées ou complexes, ont pour point commun d'incorporer à minima dans leurs processus un composant chargé de produire la structure finale de la phrase et dont le champ d'action est plus ou moins large. On peut identifier plusieurs méthodes fonctionnelles qui permettent de faire fonctionner ce composant et que nous décrivons maintenant.

1.3.1 Les générateurs à base de patrons

Les systèmes de GAT sont parfois construits d'après un ensemble de modules dédiés à une application particulière, puis assemblés dans une perspective exclusivement ori-

7. Cette architecture est décrite de manière détaillée dans (Reiter et Dale, 2000) page 48-49.
8. Dans la section 12.3.2.

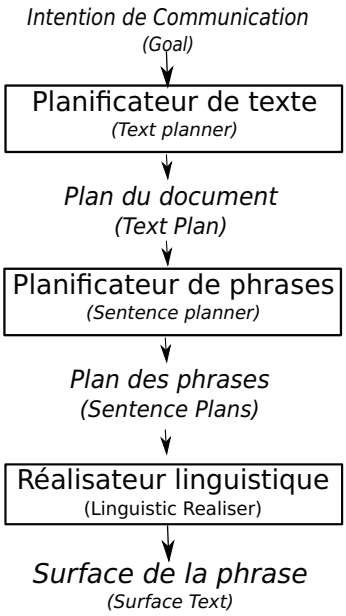

FIGURE 1.5 – *L'architecture consensuelle selon (Reiter et Dale, 2000) (avec entre parenthèses les descriptions indiquées en anglais dans l'illustration originale).*

entée vers le résultat sans réel souci de fondements théoriques. On considère souvent, à tort, que les systèmes de GAT à base de patrons entrent dans cette catégorie parce qu'ils sont simples à déployer et que leurs mécanismes sont intuitivement compréhensibles. Les systèmes de génération à base de patrons sont ceux qui introduisent directement les contenus de nature non linguistique dans la structure de surface d'une phrase [9]. Le générateur à base de patrons a donc ceci de particulier qu'il se trouve être à la fois une architecture et un composant de génération. On parle de patrons quand les structures linguistiques de surface contiennent des emplacements vides à remplir. Une phrase sera dite «générée d'après un patron» lorsque ses emplacements vides seront tous complets. Pour illustrer cette méthode, considérons une fonction f, dont la finalité est de verbaliser un itinéraire avec pour paramètres (n =numéro, d =destination, h =heure de départ).

$$f(n, d, h) = \text{Le train [n] à destination de [d] partira à [h]} \tag{1.1}$$

$$f(n = 755, d = Paris, h = 12h) \tag{1.2}$$

Dans un modèle à base de patrons, la phrase résultante de la fonction correspondra à un modèle de structure de surface (voir 1.1) dans lequel on aura procédé à l'insertion des paramètres n, d, h (voir 1.2). Ce qui donne finalement la phrase :

```
Le train 755 à destination de Paris partira à 12h.
```

9. Lire notamment (Reiter et Dale, 1997), page 83-84 sur ce sujet.

On observe que dans cette architecture rudimentaire qui correspond à celle d'un *automate à états finis* déterministe, le patron rencontre des situations où il n'est pas utilisable. La fonction 1.1 ne peut être mise en œuvre que lorsque l'heure de départ est proche ; le patron n'est pas adaptable par exemple pour un départ le lendemain. On voit également que certains cas particuliers doivent être traités (si $d = Avignon$ par exemple). Les modèles à base de patrons sont contraignants et ne peuvent pas prendre en compte des notions de temps futur ou passé puisque la structure grammaticale qu'ils produisent est théoriquement figée.

Les systèmes à base de patrons sont marginaux dans la littérature. Pour (Deemter et al., 2005) il y aurait une tendance à répartir les systèmes de génération automatique de texte en deux catégories : d'un côté les systèmes fonctionnels et mis au point pour répondre à un besoin, mais au prix d'un manque de fondements théoriques, et de l'autre des systèmes avec des fondements théoriques forts (qui reposeraient par exemple sur une théorie linguistique). Selon Deemter, il est clair que les systèmes à base de patrons sont rangés dans la première catégorie, ce qui les rend marginaux et sans intérêt scientifique. En pratique, le principal livre dédié à la GAT (Reiter et Dale, 2000), ne laisse aucune place aux systèmes à base de patrons. Cette désaffection est souvent argumentée. Les mêmes auteurs affirment ainsi dans (Reiter et Dale, 1997) que les systèmes à base de patrons sont plus délicats à maintenir et à mettre à jour et qu'ils produisent des sorties plus pauvres et moins variables [10] que les systèmes de GAT dits «standards» (c'est-à-dire reposant sur un générateur de phrases à base de règles, inséré dans une architecture *pipeline*). (Busemann et Horacek, 1998) poussent plus loin ce raisonnement en suggérant que les systèmes à base de patrons ne sont intrinsèquement pas capable d'incorporer des compétences linguistiques [11]. Ces auteurs affirment que les techniques à base de patrons manquent de flexibilité et sont quasiment inutilisables en tant que composants de génération de phrases d'un système complexe [12]. Pourtant, ainsi que le remarque (Reiter, 1995), il existe probablement des millions d'applications qui utilisent des patrons plutôt que des méthodes linguistiques pour produire du texte de manière mécanisée. Pour Reiter, ces systèmes qu'il regroupe sous le vocable de *Automatic Text Generation* (ATG) pour les distinguer de ceux de la famille des *Natural Langage Generation* (NLG) représentent probablement plus de 99,9% des systèmes de GAT !

1.3.2 Les générateurs à composants à base de règles de production

Par tradition, les systèmes de GAT ont été conçus comme des systèmes de décision déterministes. Ceci signifie qu'ils consistent bien souvent en un processus dans lequel on introduit un ensemble de mots, plus ou moins structuré, et qu'au fil de transformations successives marquées par des décisions la plupart du temps pré-déterminées (c'est-à-dire programmées par le concepteur du système), cet ensemble de mots va

10. (Reiter et Dale, 1997), p60, 61 et 84.
11. (Busemann et Horacek, 1998) p 238.
12. «*However, the benefits of both surface realization components and template-based techniques are still limited from an application-oriented perspective. Surface realization components are difficult to use because of the differences between domain-oriented and linguistically motivated ontologies (as in SPL), and existing template-based techniques are too inflexible*» dans (Busemann et Horacek, 1998) p238.

progressivement se transformer en phrase. En pratique, on utilise pour produire les phrases, des générateurs qui implémentent une théorie linguistique. Ces systèmes minutieusement mis au point sont souvent composés de règles largement écrites manuellement, associés à des composants de génération (gestion des temps, des procédés anaphoriques) qui prennent de nombreuses décisions localement. Cette façon de concevoir la génération, largement dominante, n'est pas sans conséquence. De tels générateurs sont soit exagérément généralistes soit exagérément spécialisés dans un domaine et délicats à adapter pour de nouvelles applications. Ils souffrent également d'une très forte dépendance à leur langue de conception qui les rend très difficiles à adapter dans un autre langage. (Reiter et Dale, 2000) décrivent exclusivement cette famille de systèmes.

1.3.3 Les générateurs à composants statistiques et apprentissage sur corpus

(Belz, 2005) souligne que la domination de ces systèmes à base de règles explique peut être pourquoi la GAT demeure un champ de recherche sous-représenté dans le domaine du TAL. Alors que les tâches du TAL sont aujourd'hui servies par un ensemble de méthodes robustes, adaptables, fiables et éprouvées, la GAT souffre selon Belz de sa méthodologie. Une réponse alternative entre l'absence de règles linguistiques dans les systèmes à base de patrons et celle des systèmes classiques réside dans l'introduction de connaissances issues de l'observation de corpus par des approches statistiques. Les connaissances ainsi obtenues sont généralement exploitées dans les composants de planification et de réalisation surfacique des systèmes de GAT (comme par exemple dans le système FERGUS (Chen et al., 2002)). Ces systèmes sont souvent déployés dans des contextes de dialogue pour renforcer la robustesse d'un système de génération rudimentaire à base de patrons sans avoir à implémenter les composants de production linguistique, ou pour limiter le plus possible le travail de conception de ces composants. Ainsi, l'approche de (Langkilde et Knight, 1998a) dans le système NITROGEN consiste à utiliser des n-grammes extraits d'un corpus en sélectionnant les plus appropriés d'après le meilleur chemin et à appliquer les séquences sur la production issue d'un modèle à base de patrons. On observe un regain d'intérêt pour les méthodes statistiques, avec l'insertion de modèles n-grammes guidés par des grammaires comme dans (Guo et al., 2008) ou encore, plus récemment, à travers les propositions présentées dans (Krahmer et Theune, 2010), définies comme *approches de générations texte vers texte*. Ces systèmes à apprentissage automatique innovent en introduisant une séparation entre la définition de l'espace de génération (ce qui permet de passer de l'intention à la phrase produite) et le contrôle des transformations opérées dans cet espace. Le système statistique est donc beaucoup moins contraint que celui à base de règles et ainsi plus riche et adaptable. Il est aussi susceptible de produire plus facilement des erreurs.

1.3.4 Représentativité des différentes approches

L'inventaire le plus détaillé disponible sur les systèmes de GAT, initialement publié dans (Adorni et Zocks, 1993), est entretenu par John Bateman et Michael Zock. Il est

mis à jour à travers des publications en ligne [13]. Une analyse de cette ressource nous permet de mesurer le succès des différentes propositions qui viennent d'être décrites.

- **Systèmes à base de patrons** : une dizaine de systèmes à base de patrons sont référencés. En réalité, ce modèle théorique est probablement le plus répandu et déployé de tous. On le rencontre fréquemment en tant que composant de production de texte dans des systèmes de dialogue (voir par exemple (Benamara, 2004)).
- **Systèmes à base de grammaires et de règles** : bien qu'ils ne soient ni les plus répandus (les systèmes à base de patrons sont incontestablement les plus nombreux) ni les plus utilisés dans des applications industrielles opérationnelles, ils sont très largement majoritaires dans la littérature scientifique. Bateman et Zocks référencent les applications qui ont fait l'objet de recherches approfondies et de publications récurrentes. Si on fait abstraction des grammaires très variées qu'ils implémentent (génératives, à base d'arbres,voir chapitre 2), on peut considérer que 70 systèmes sur la centaine présentés sont regroupés dans cette catégorie.
- **Systèmes à composants statistiques** : les systèmes dont le composant de production de formes de surface des phrases utilise au moins en partie une méthode statistique sont les moins nombreux. Au plus une dizaine si l'on considère tous les composants qui entrent en jeu (y compris ceux dont le rôle est mineur mais faisant appel à un corpus pour la correction des phrases).

1.4 Conclusion

L'architecture interne des systèmes de GAT a donné lieu a de nombreuses propositions dont le principal point commun est la modularité au sein d'un système *pipeline*. Il semble faire consensus qu'un système de GAT doit appliquer à une représentation abstraite d'une *Intention de Communication* un ensemble de traitements successifs, afin de transformer cette intention en phrase. Selon les propositions, les étapes sont plus ou moins nombreuses et l'intrication de la problématique de la gestion du plan avec celle de la génération de phrases (même si elle n'est pas sans fondement, par exemple dans la perspective de traiter les anaphores), rend parfois la lisibilité des architectures difficile. Les systèmes de GAT à base de règles et de grammaires organisés selon une architecture *pipeline* sont normalisée de facto depuis les années 60, et laissent pour le moment peu de place aux alternatives. Nous verrons qu'il résulte de cette restriction du champ d'investigation scientifique de nombreuses questions ouvertes. La plupart des systèmes de référence sont monolingues. Leur adaptation, faute de méthode de construction partiellement ou complètement automatique est longue, complexe et coûteuse. On notera l'étude de (Deemter et al., 2005) qui tend à démontrer que les architectures à base de patrons peuvent s'hybrider avec profit avec une architecture *pipeline*. Dans le registre des approches statistiques, on mentionnera aussi les travaux récents de (Belz, 2006; Krahmer et Theune, 2010) qui tendent à renouveler les architectures de GAT de nature statistique et à base de corpus.

13. http://www.fb10.uni-bremen.de/anglistik/langpro/NLG-table/NLG-table-root.htm puis sur http://www.nlg-wiki.org/

Chapitre 2

Générateurs à base de règles et de grammaires

Sommaire

Les systèmes de GAT dominants dits *à base de grammaires* sont composés de modules assurant la transformation par étapes successives d'une *Intention de Communication* en une phrase. Ces éléments modulaires sont agencés au sein d'une architecture qui les active successivement en utilisant à chaque étape les transformations réalisées lors des étapes précédentes. Les composants directement impliqués dans la génération finale de la phrase font généralement appel à des règles de production grammaticales qui assurent la tâche de transformation d'une *Intention de Communication*, au besoin lexicalisée. Ce sont les composants de *planification de phrase* et de *réalisation de surface* qui s'acquittent de cette tâche. La mise en forme de *l'Intention de Communication*, la gestion de la *lexicalisation* et la *gestion anaphorique* seront laissées de côté dans ce chapitre car faisant généralement appel à des méthodes non linguistiques.

2.1 Théories linguistiques impliquées

Qu'ils exploitent des grammaires génératives (Chomsky), hybrides (Harris), dérivées de la théorie Sens-Texte (Igor Mel'cuk), voire encore à base de grammaires d'arbres adjoints (TAG) (Joshi), tous les systèmes que nous décrivons dans ce chapitre revendiquent leur appartenance à une famille théorique de la linguistique. Cette appartenance a une

influence importante sur la conception des systèmes, leur adaptation d'un langue à une autre et sur leur simplicité d'usage. Nous passerons donc ici en revue les approches théoriques revendiquées par les système de GAT et tenterons de souligner les conséquences de leur utilisation.

2.1.1 Grammaires génératives et transformationnelles

La grammaire générative est la théorie linguistique proposée par Chomsky à partir de 1955 selon laquelle un ensemble de règles grammaticales permettent de générer toutes les phrases de la langue et par extension qu'il existe des règles communes à toutes langues. A partir de 1955 et malgré des modifications ultérieures, les grandes lignes de ce qui est devenu la grammaire générative et transformationnelle (partie de la grammaire générative que Chomsky a étudiée) sont posées. Cette théorie ne résout pas le problème de l'origine et de la nature des langues d'un point de vue scientifique mais propose une nouvelle étude linguistique. La théorie de Chomsky joue un rôle essentiel dans la GAT car elle est l'une des premières à faire l'objet de mise en œuvre dans des applications de génération, comme nous l'avons indiqué dans l'introduction. Elle est implémentée dans le système de génération automatique de phrases aléatoires de (Yngve, 1961), puis dans le système d'analyse de phrases artificielles de (Matthews, 1962). Elle joue à ce titre un rôle historique.

La grammaire générative entend expliquer comment le langage permet à un individu de comprendre et de produire des phrases qu'il n'a jamais entendues. Pour atteindre cet objectif, elle distingue l'activité linguistique réelle du sujet, de sa compétence, c'est à dire sa connaissance implicite de la langue. Fondamentalement, la grammaire générative s'attache à décrire et expliquer la compétence linguistique. Son postulat initial est qu'un ensemble fini de règles pré-existe dans le système cognitif de l'individu parlant, et que cet ensemble lui suffit pour engendrer un nombre infini de phrases. On voit immédiatement l'intérêt de ce postulat, qui permet d'ailleurs dès les années 60 d'implémenter avec des moyens informatiques peu puissants des règles de GAT en langage Fortran. La théorie distingue structure de surface (la phrase) et structure de profondeur (proche du concept d'*Intention de Communication*). Une telle proposition a rendu nécessaire l'introduction de mécanismes permettant d'expliquer le passage d'une structure profonde abstraite à une structure de surface : ce sont les transformations, d'où le terme de grammaire transformationnelle.

2.1.2 Implémentation dans les systèmes de GAT

La théorie chomskyenne a vécu de nombreuses transformations au cours de son histoire, que ses transformations soient intervenues sous l'action de son auteur ou celle de schismes. Ces différentes variations se retrouvent dans la diversité des appartenances revendiquées par les systèmes de GAT. Les implémentations de ces règles de production et de transformation dans des systèmes de GAT répondent donc à de nombreuses propositions.

- **SFG** ou *Systemic functional grammar* [1] : un modèle de grammaire qui fait référence au langage en tant que réseau de systèmes où le terme fonctionnel décrit une approche pratique et conceptualisée des assemblages d'objets textuels pour obtenir du sens, par opposition aux grammaires formelle, plus axées sur la syntaxe et les classes de mots. Le système PENMAN (Matthiessen et Bateman, 1991) est la plus connue des nombreuses implémentations (29 recensées sur nlg-wiki).
- **LFG** ou *Lexical Functionnal Grammar* [2] : ce modèle grammatical a été initié par Joan Bresnan and Ronald Kaplan dans les années 70 en réaction aux orientations de recherche prises dans le domaine des grammaires transformationnelles. Ce modèle est principalement axé vers la syntaxe et sa relation avec la morphologie et la sémantique. Le système de GAT CHARON (Arnold et al., 1993) utilise les grammaires LFG.
- **TAG** ou *Tree-Adjoining grammar* [3] : il s'agit d'un formalisme défini par (Joshi et Rambow, 2003) dans la perspective des grammaires d'adjonction de Zellig Harris. Les grammaires TAG sont proches des grammaires hors contexte mais les règles élémentaires de ré-écriture recourent à un arbre et non plus à un symbole. Les grammaires TAG utilisent des règles pour ré-écrire les noeuds des arbres sous forme de nouveaux arbres. De nombreux systèmes (voir par exemple (Stone et Doran, 1997; Meunier et Danlos, 1998; Danlos et al., 2000)) reposent sur cette grammaire qui offre l'avantage de gérer de manière élégante le problème de la relation de sens dans une phrase.
- **GBG** ou *Government and Binding Theory (Théorie du liage)* : il s'agit d'une proposition d'application à la GAT des principes et des modèles théoriques développés par (Chomsky, 1969, 1957). Ce nom fait référence aux deux sous-théories dérivées de la proposition originale : le *gouvernement* qui est une relation syntaxique abstraite et la *liaison (binding)* qui traite de la relation avec les pronoms, les anaphores et les expressions co-référentes. Le système GB Gen (Etchegoyhen et Wehrle, 1998) est une des implémentations connues.

2.1.3 Théorie Sens-texte

La première publication présentant les fondements de la *Théorie Sens-Texte* (TST) remonte à plus de quarante ans (Mel'čuk et A, 1970). La TST rend compte de l'association que tout locuteur d'une langue L est capable de faire entre un sens donné de L et l'ensemble des énoncés paraphrastiques de L exprimant ce sens. La TST considère donc la langue comme une machine virtuelle permettant de traduire des Sens en énoncés, appelés Textes, et vice-versa. On devra prendre soin de considérer que la machine virtuelle de la TST ne permet pas de produire l'ensemble infini des énoncés grammaticaux d'une langue (approche de type chomskyenne). Ce caractère non génératif de

1. Des informations sur SFG sont disponibles sur http://www.isfla.org/Systemics/Definition/definition.html.
2. Le groupe de support de LFG présente ses travaux sur http://www.essex.ac.uk/linguistics/external/LFG/.
3. On peut se reporter sur le projet X-TAG pour plus d'informations http://www.cis.upenn.edu/~xtag/, qui est historiquement la première implémentation d'une TAG.

la TST est d'ailleurs un de ses points forts. Elle lui permet de ne privilégier aucune description de langue en particulier lorsque son caractère trop proche de la langue anglaise l'un des principaux reproches fait à la théorie chomskyenne.

Autre différence importante par rapport à l'approche chomskyenne, bien que la TST repose sur des principes généraux universaux, la détermination de ces universaux n'est pas considérée comme une fin en soi. La finalité de la théorie est, au-delà de la découverte des universaux linguistiques, la construction des modèles particuliers de chaque langue. La TST est donc orientée vers la description et se veut un outil pour le lexicographe et le grammairien. Un modèle linguistique Sens-Texte consiste avant tout en un lexique, une grammaire et un ensemble de procédures permettant d'activer ces deux composantes pour effectuer la connexion entre Sens et Texte, symbolisée par la formule Sens - Texte. Pour résumer, la TST est une théorie linguistique visant la description de la correspondance entre Sens et Texte au moyen de la construction de modèles formels. Ces modèles peuvent être considérés comme des machines logiques virtuelles. Ces machines virtuelles prennent en entrée des représentations et retournent en sortie un ensemble de paraphrases permettant d'exprimer le Sens donné en entrée.

Implémentations de la TST dans des systèmes de GAT

La théorie proposée par Igor Mel'cuk s'adapte particulièrement à la problématique de la GAT. Elle est implémentée dans plusieurs systèmes de GAT et de paraphrasage (sept inventoriés depuis le milieu des années 80). L'implémentation de la TST la plus convaincante en GAT nous semble être celle du système REALPRO telle qu'elle est décrite dans (Lavoie et Rambow, 1997). La TST, formellement, conçoit la production de texte comme un processus avec 7 niveaux de représentation. Lavoie et Rambow respectent cette hiérarchie et la terminologie de (Mel'čuk, 1988). Ils décrivent leur système comme recevant en entrée une *structure de dépendance syntaxique* (DSyntS). Les DSyntS sont organisés sous forme d'arbres dont les noeuds et les branches sont étiquetés. Le système fonctionne à un niveau lexical profond en utilisant des lexèmes [4]. Les arbres de dépendance qui représentent un DsyntS ne contiennent pas de noeuds non-terminaux et tous les noeuds sont étiquetés par un lexème. Les arcs qui relient les noeuds sont tous étiquetés par des étiquettes syntaxiques telles que *Sujet (I)*.

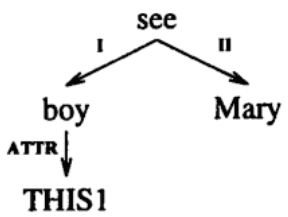

FIGURE 2.1 – *L'arbre de DSynts de REALPRO avant sa linéarisation en phrase.*

4. Au sens de mots non fléchis.

La représentation contenue dans un DSyntS est strictement de nature syntaxique profonde : elle n'inclut aucun mot fonctionnel et seuls les lexèmes jouant un rôle sémantique sont décrits. On voit un exemple applicatif dans l'arbre décrit dans la figure 2.1, qui correspond à la phrase :

```
This boy sees Mary
```

Le principe fonctionnel de REALPRO repose sur une architecture *pipeline*. Dans un premier temps la validité syntaxique des DSyntS est vérifiée. Puis DSyntS est transformée pour recevoir la forme prévue par une règle par qui lui correspond. Un composant de gestion de la structure syntaxique profonde prend alors en entrée le DSyntS modifié. La grammaire DSyntS et le lexique sont alors sollicités pour insérer les mots fonctionnels (tels que les auxiliaires, les prépositions) et produire un nouvel arbre de dépendance. Ce nouvel arbre inclut les structures SSynt et des arcs dont les labels sont plus précis. Le composant de *réalisation de surface* transforme alors les noeud du SSynt en une forme linéaire proche de la phrase finale. Ce module prend en charge la structure morphologique et utilise la grammaire SSynt pour gérer le cheminement linéaire de la phrase en prenant en compte les labels apposés sur l'arbre par le composant de gestion de structure syntaxique profonde. Finalement, le composant morphologique profond ajoute les flexions des éléments de DMorphS en transformant les formes de surfaces à l'aide de structures morphologiques (les SMorphS). Les informations morphologiques utilisées par DMorphS sont issues du lexique. Dans un dernier composant, les ponctuations et le formatage final de la phrase sont mis en place.

2.2 Composants autonomes de génération de surface

Nous venons de présenter sous diverses implémentations issues d'une théorie, le module le plus caractéristique d'une application de GAT. À savoir celui qui se charge de la *réalisation linguistique* en produisant la *forme de surface* finale d'après une structure construite lors des phases de préparation de *l'Intention de Communication* et de *planification de phrase*. Des propositions ont été faites pour tenter de mettre à disposition des chercheurs et des industriels des composants de *réalisation linguistique* ouverts et prêts à l'emploi. SimpleNLG décrit dans (Reiter et Gatt, 2009) [5] ; KPML (Bateman, 1997) [6], le plus ancien composant développé dans les années 80, qui propose des grammaires dans 10 langages ; FUF/SURGE (Elhadad et Robin, 1996) [7] mis au point dans les années 90, toujours en usage dans de nombreux projets, sont quelques exemples de proposition de composants normalisés. On notera, à nouveau, que les multiples dénominations employées pour une même catégorie d'applicatif ne sont pas toujours aisées à maîtriser pour le néophyte : Elhadad parle pour FUG de *syntactic realization component*, Bateman décrit KPML comme un *Blackbox generator* ou encore un *environnement de développement*, Gatt de *realisation engine*, tous ces auteurs faisant à un moment ou un autre référence à

5. Téléchargeable sur http://www.csd.abdn.ac.uk/research/simplenlg/.

6. Disponible sur le site http://www.fb10.uni-bremen.de/anglistik/langpro/kpml/README.html.

7. L'URL de téléchargement de FUF n'est malheureusement plus disponible.

(Reiter, 1994) qui lui même parle de *linguistic realization* dans son ouvrage académique sur la génération de texte [8]. Tous ces termes ne décrivent qu'un seul et même module stratégique dont la tâche dévolue au sein d'une architecture de GAT ne concerne que la transformation d'une *Intention de Communication* - généralement lexicalisée - en *forme de surface de phrase*.

Dans tous les cas, les auteurs s'accordent à considérer que la *génération de formes de surfaces* s'inscrit dans une architecture consensuelle telle que décrite en 1.2 à savoir composée de trois modules qui se chargent par étapes successives de transformer l'*Intention de Communication* en *phrase générée* en passant par (i) *la définition de contenu* (ii) la *planification de phrases* (iii) la *génération de surface*.

SimpleNLG

SimpleNLG (Reiter et Gatt, 2009) appartient à cette catégorie de *générateur de surface* réutilisables pour systèmes de GAT et s'inscrit au sein de l'architecture consensuelle. SimpleNLG fonctionne exclusivement pour la langue anglaise. Cette librairie logicielle illustre à la perfection ce que pourraient être ces composants modernes et simples à déployer dans des applications de GAT que (Belz, 2005) appelle de ses vœux et présente comme susceptibles de favoriser le développement de la recherche et du domaine applicatif de la GAT. Exactement comme des logiciels largement diffusés de classification ou d'étiquetage ont favorisé en leur temps le développement du TAL. Pour construire une structure syntaxique et linéariser cette dernière sous forme d'une phrase, SimpleNLG demande quatre étapes :

1. Dans un premier temps, il faut initialiser les constituants de base de la phrase en utilisant les éléments lexicaux appropriés.
2. L'*Application Programming Interface* (API) de SimpleNLG va ensuite permettre de déclarer des propriétés complémentaires tels que le temps des verbes, le mode (interrogatif, affirmatif) ou encore la forme (passive, active), destinées à la future structure syntaxique .
3. En combinant ces informations, une structure plus formelle va être élaborée par l'API.
4. Pour finir, la structure résultante est transmise à un réalisateur de surface intitulé *lineariser* qui applique aux constituants les transformations qui doivent résulter de l'application des propriétés (vues au point 2). La phrase sous sa forme finale est alors retournée.

On notera que les conjonctions de la famille des connexions logiques telles que *and*, *or* sont gérées implicitement par le logiciel lorsqu'il concatène plusieurs structures de constituants. Si nous reprenons les 4 étapes ci-dessus, la construction d'une phrase dans SimpleNLG peut être décrite selon ce qui suit. Dans un premier temps, les éléments lexicaux sont déclarés en utilisant une structure de type sujet, verbe et complément. Les propriétés telles que les temps ou les formes ne sont pas déclarées à ce stade (voir le code présenté dans 2.1).

8. Voir (Reiter et Dale, 2000) p57.

```
SPhraseSpec p = new SPhraseSpec();
p.addSubject("my dog");
p.addSubject("your cat");
p.addSubject("the tiger");
p.addSubject("the monster");
p.setVerb("chase");
p.addComplement("George");
```

TABLE 2.1 – *Définition des éléments lexicaux d'une phrase avec SimpleNLG.*

Dans un second temps les propriétés sont déclarées, comme dans le code décrit dans 2.2, où le temps est positionné sur le passé.

```
p.setTense(Tense.PAST);
p.setProgressive(true);
p.setNegated(true);
p.setPassive(true);
```

TABLE 2.2 – *Définition des propriétés d'une phrase avec SimpleNLG.*

Pour finir, la forme est construite, les transformations sont appliquées (étape 3 et 4), en utilisant le code décrit dans 2.3.

```
Realiser r = new Realiser();
String output = r.realiseDocument(p);
System.out.println(output);
```

TABLE 2.3 – *Transformation et réalisation d'une phrase avec SimpleNLG.*

Et la phrase suivante est retournée :

```
George was not being chased by my dog, your cat, the tiger and
the monster.
```

Le composant lexical et le composant syntaxique sont implémentés sous forme de règles pré-définies en langage Java et répondent aux normes suivantes :

- Le composant lexical fournit une interface permettant de définir un lexique, des règles morphologiques et des éléments lexicaux complémentaires. Les règles de transformation morphologiques sont implémentées d'après MORPHG (Minnen et al., 2001), et couvrent la totalité des flexions de la grammaire anglaise, incluant les formes régulières et irrégulières. Des expressions régulières permettent de déterminer la forme de n'importe quel verbe.
- Le composant syntaxique de SimpleNLG contient une interface pour les noyaux de syntagmes (HeadedPhrases) et les coordinations de syntagmes (CoordinatePhrase). Les sous-types de syntagmes suivent les usages définis par (Reiter et Dale, 2000). Des règles de temps, numérales et personnelles permettent d'attribuer une propriété.

Bon exemple de ce que peut être un composant de génération standardisé facile à déployer, SimpleNLG est aussi une parfaite illustration du principal reproche qui pourrait être fait à ces systèmes dont l'architecture n'a guère variée depuis les années 60 : ils sont extrêmement difficiles à adapter à une autre langue que celle de leur conception. Près des deux tiers des classes de SimpleNLG contiennent des règles de transformations spécifiquement conçues pour l'anglais et la morphologie de ses verbes. Nous verrons en revanche dans la partie III que le tiers de classes restant constitue un ensemble de fonctions adaptables à toute langue de typologie *Sujet, Verbe, Objet* (tel que le français, l'anglais ou l'espagnol) que nous réemploierons.

2.3 Conclusion

Les grammaires utilisées dans les systèmes de GAT qu'elles soient hors contexte, transformationnelles, issues de la théorie Sens Texte, à base d'arbres, ont toutes en commun qu'elles associent nécessairement un ensemble de connaissances linguistiques manuellement implémentées aux processus de transformation d'une *Intention de Communication* en phrase. Il ne suffit pas d'implémenter les algorithmes relevant d'une théorie pour transformer une *Intention de Communication* en structure phrastique lorsque l'on utilise un arbre ou une règle de transformation. Il faut aussi introduire dans le système des connaissances grammaticales telles que les formes des verbes conjuguées selon leur temps, les exceptions linguistiques. De nombreuses grammaires ont été expérimentées. Parfois une théorie grammaticale isolée n'est implémentée que dans un seul système. Le principal avantage de cette famille de composants de GAT est leur haut niveau de performance (ils génèrent des phrases instantanément) leur grande fiabilité (ils ne produisent quasiment jamais d'erreurs syntaxiques et sémantiques) et leur simplicité de déploiement. En contrepartie, ils sont difficiles à adapter dans une autre langue que celle de leur conception (le plus souvent l'anglais) et présentent une pauvreté stylistique de sortie marquée. Pour finir, ces composants de GAT de production grammaticale sont encore peu normalisés ce qui freine leur développement.

Chapitre 3

Méthodes statistiques appliquées à la génération automatique de texte

Sommaire

Nous avons souligné en étudiant les architectures existantes (1.3.4) qu'il existait une approche dominante de la GAT. Cette approche prévoit une architecture de type *pipeline* à trois composants dits *tactiques*. Ces *composants tactiques* mettent en forme une *Intention de Communication* avec un *Gestionnaire de l'intention de communication*, un *planificateur de phrase* et un *Générateur de formes de surface* qui produit la phrase terminale. Des méthodes logiques assurent la préparation de l'*Intention de Communication* avant sa transformation en phrase par un *Générateur de forme de surface* à base de règles inspirées d'une théorie linguistique. On observe que tous ces éléments d'un générateur de texte pourraient à un moment ou un autre faire appel à une méthode statistique, exploiter des applications d'extraction d'information ou d'apprentissage appliquées à un corpus. Ces méthodes ont démontré dans de nombreux domaines du TAL (étiquetage, compréhension sémantique ou syntaxique) leur aptitude à gérer de manière performante des tâches complexes aussi bien que les systèmes à base de règles. Cependant comme le rappelle (Belz, 2006), *«sur les 30 systèmes et modules [de GAT] implémentés depuis ou après 2000, listés sur le site de référence de la GAT* [1]*, seulement 5 sont munis de composants statistiques (et six autres mettent en œuvre des méthodes qui d'une manière ou d'une autre font appel à des corpus)»*. L'auteure en conclut qu'il est possible que l'une des raisons probables de ce désintérêt pour les systèmes de GAT probabilistes soit *«[que] la plupart des*

1. Liste de Bateman et Zocks.

techniques de GAT reposant sur des méthodes statistiques sont coûteuses par nature, exigeant que les ensembles d'alternatives [phrastiques] à produire soient intégralement générés avant de pouvoir être [exploités] par un modèle statistique». Elle envisage qu'il faille considérer plus simplement que «*les méthodes de génération de texte à base de statistiques n'ont peut être pas encore démontrées leur capacité à produire des sorties d'une qualité suffisante*».

La question de la nature générative[2] d'un alphabet ou d'un lexique est explorée méthodiquement depuis Leibnitz et, dans le cadre particulier de la linguistique computationnelle, par Shannon dans sa *Théorie mathématique de la communication* (Shannon, 1948). Il fût le premier à tenter - dans une perspective d'amélioration des communications - d'exploiter les capacités d'un système combinatoire probabilisé de l'alphabet ou d'un lexique à modéliser des séquences de lettres (pour reproduire des mots) ou des séquences de mots (pour reproduire des phrases). Dès les années 50, et sans capacité de calcul, Shannon en utilisant les processus markoviens et des probabilités de transitions d'un mot à un autre manuellement recensées dans un corpus, était en mesure de produire aléatoirement des phrases syntaxiquement correctes. Il utilisait pour cela un lexique et une approximation de deuxième ordre (c'est-à-dire prenant en compte la probabilité d'apparition d'un mot d'après son prédécesseur) comme dans cet exemple qu'il fournit en préambule de l'article énonçant sa théorie :

```
THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH WRITER THAT THE
CHARACTER OF THIS POINT IS THEREFORE ANOTHER METHOD FOR THE
LETTERS THAT THE TIME OF WHO EVER TOLD THE PROBLEM FOR AN
UNEXPECTED.
```

Shannon observa qu'avec cette méthode, «*la ressemblance avec un texte en anglais ordinaire s'améliore* [*en fonction du degré de complexité*][...] *et que les exemples* [*ainsi produits*] *offrent une relativement bonne structure, au delà de la portée* [*cad du degré d'ordre NdEC*] *qui a été prise en compte pour leur construction*». En d'autres termes, Shannon démontrait la capacité de la propriété de Markov à modéliser le langage et en particulier sa structure syntaxique par le simple inventaire des probabilités de co-occurrences des objets textuels qui le composent. A savoir qu'un processus stochastique dont la distribution conditionnelle de probabilité des états futurs, étant donné les états passés et l'état présent, ne dépend en fait que de l'état présent et non pas des états passés (absence de «mémoire»). Ces exemples illustrent à quel point on cherche à savoir depuis longtemps si les systèmes statistiques ont un réel potentiel de génération.

Les méthodes de GAT existantes reposant sur des approches statistiques exploitent deux grandes familles de techniques. Elles utilisent des observations statistiques faites sur des corpus pour guider des systèmes de génération de texte pendant la phase conceptuelle (Varges et Mellish, 2001; White, 2004). Ou bien elles recourent à des modèles *n*-grammes pour générer une forme de surface plus riche ou plus appropriée postérieurement et en complément d'un processus de GAT déjà réalisé (Knight et Hatzivassiloglou,

2. Par *nature générative* nous entendons, la capacité d'un ensemble d'éléments à produire par association un système d'ordre supérieur complet. Ainsi, la combinaison des lettres d'un alphabet est capable de générer la totalité des mots d'une langue, de même que des combinaisons lexicales peuvent reproduire toutes les phrases d'un langage.

1995; Langkilde et Knight, 1998b; Bangalore et Rambow, 2000b). Il s'agit ici plus d'une étape de réécriture et de correction grammaticale que de production de la forme de surface (on reviendra plus loin sur ce point en décrivant les logiciels de la famille HALOGEN). On ajoutera - bien qu'il s'agisse d'une problématique différente de celle traitée dans ce chapitre - que les ressources lexicales et conceptuelles qui peuvent être impliquées dans un système de GAT sont elles aussi concernées par des méthodes d'apprentissage statistique parfois proches de celles impliquées dans la génération. Mentionnons pour l'exhaustivité une approche intermédiaire utilisant un processus d'extraction d'information faiblement numérique mais très original (Kosseim et al., 2001) élaboré en vue de répondre automatiquement à des emails.

Nous ajoutons que des recherches parallèles ont envisagé la possibilité que le processus de génération puisse être considéré comme un problème de satisfaction de contraintes, étudié par les méthodes de la *Recherche Opérationnelle*. Ce point de vue mérite d'être mentionné ici puisqu'il considère le caractère hautement combinatoire du langage comme un problème mathématique à résoudre. Des précurseurs en ce domaine ont été le système PAULINE (Hovy, 1988) qui utilise pour satisfaire les contraintes des objectifs rhétoriques ou encore ICONOCLAST (Power, 2000) qui autorisait l'utilisateur du système de GAT à régler finement des combinaisons de contraintes. On mentionnera plus récemment les travaux de (Hankash, 2009) qui explorent cette approche. Ici, en plus de la définition de contraintes pour la cohérence du texte, est formulé un ensemble de contraintes qui permet de façonner l'IC [3] en fonction des buts communicatifs afin de favoriser leur réalisation. Il est également proposé une solution au problème de la complexité de calcul de la génération de textes de grande taille. Nous ne nous attarderons pas sur ces approches particulières du problème de la GAT. Nous nous contenterons de noter qu'il existe une communauté de vue entre l'idée de modéliser les probabilités de co-occurences de mots dans un corpus utilisé en tant que support d'apprentissage et un système de GAT par contraintes qui cherche à formaliser de manière mathématique ces contraintes. On se trouve ici à un stade intermédiaire hybride entre les systèmes à base de règles et les systèmes à base de statistiques.

3.1 Principes théoriques de base des approches statistiques

Les modèles de génération statistiques font majoritairement appel à une sélection de mots par n-grammes et à des modèles de langage. Dans un modèle n-grammes, la probabilité $P(w_1, \ldots, w_n)$ d'observer une phrase composée des mots $w_1, ..., w_n$ est estimée par le produit des probabilités d'apparitions individuelles des mots contenus dans la séquence (formule 3.1).

$$P(w_{1,n}) \approx P(w_1)P(w_2)...P(w_n) \tag{3.1}$$

Ce modèle dit *unigramme* repose sur l'hypothèse forte que dans une suite de mots, chaque mot apparaît indépendamment et qu'en conséquence, la probabilité d'appari-

3. Le terme de macrostructure est utilisé par l'auteur.

tion d'un mot dans une séquence de n mots peut être estimée par le produit des probabilités d'apparition des n mots qui le précèdent, pris indépendamment. Il exploite la propriété de Markov d'ordre 1. Par extension, on peut utiliser une propriété de Markov d'ordre supérieur en estimant la probabilité d'apparition d'un mot non plus pris isolément, mais lui même conditionné par les mots qui le précèdent : on utilise alors un modèle n-grammes, c'est-à-dire le plus souvent 2 (bigrammes) ou 3 mots (trigrammes). On aura ainsi pour un modèle bigramme appliqué à une séquence de n mots w la formule 3.2.

$$P(w_1, \ldots, w_n) \approx P(w_1)P(w_2|w_1)...P(w_n|w_{n-1}) \tag{3.2}$$

Pour un modèle trigramme appliqué à une séquence de n mots (formule 3.3).

$$P(w_1, \ldots, w_n) \approx P(w_1)P(w_2|w_1)P(w_3|w_{1,2})...P(w_n|w_{n-2,n-1}) \tag{3.3}$$

que l'on peut généraliser pour une phrase de n mots w avec une estimation par des n-grammes de longueur n (formule 3.4).

$$P(w_1, \ldots, w_n) \approx \prod_{i=1}^{n} P(w_i|w_{i-(n-1)}, \ldots, w_{i-1}) \tag{3.4}$$

Les probabilités conditionnelles sont préalablement calculées dans une liste de n-grammes ordonnée par leur fréquence d'apparition et apprises depuis un corpus représentatif (3.5). En pratique, le *modèle de langage (ML)* n-grammes est appris avec un outil du domaine public tel que SRLIM (Stolcke, 2002).

$$P(w_i|w_{i-(n-1)}, \ldots, w_{i-1}) = \frac{count(w_{i-(n-1)}, \ldots, w_{i-1}, w_i)}{count(w_{i-(n-1)}, \ldots, w_{i-1})} \tag{3.5}$$

3.1.1 Les modèles de langage n-grammes

Le principe fonctionnel d'un composant de génération à base de méthode statistique peut donc se résumer ainsi : une représentation symbolique d'une partie d'une phrase à élaborer est produite et un *modèle de langage* (ML) à base de n-grammes est exploré pour sélectionner la réalisation qui offre la meilleure probabilité d'être syntaxiquement correcte. Les méthodes à base de modèles n-grammes ont de nombreuses propriétés appréciables dans un système de GAT. Elles sont en particulier robustes puisque le processus de sélection qu'elles induisent repose sur une observation statistique. Les ML n-grammes peuvent être appris sur des corpus dédiés et donc tenir compte du champ sémantique d'un domaine d'application (par exemple les spécificités du discours juridique ou du contenu encyclopédique). On considère qu'en théorie, un ML appris sur un corpus infini d'une langue donnée, est génératif pour cette langue (c'est-à-dire qu'il inventorie toute les formes possibles de la langue). En pratique, les ML sont appris sur

des corpus finis qui correspondent à un domaine. Le texte source utilisé pour l'apprentissage du ML devra donc être adapté au domaine applicatif recherché.

3.1.2 Implications des modèles n-grammes dans des systèmes proches de la GAT

L'une des premières applications concrètes des travaux exploratoires de Shannon fût l'approche statistique proposée pour la *traduction assistée (TA)* par ordinateur de (Brown et al., 1990). Dans le modèle proposé, on utilise le théorème de Bayes pour minimiser le risque d'erreur d'une traduction de phrases en exploitant deux corpus alignés. L'idée est que pour une phrase T dans un langage cible, il est possible de choisir une phrase S la plus probable en postulant qu'elle maximise $p(S|T)$. On notera que si l'on fait abstraction du fait que T et S sont des phrases écrites dans deux langues différentes, transformer une phrase en une autre en conservant son sens est aussi une activité de réécriture. Une autre application qui peut être considérée comme une forme de réécriture de phrase est le décodage de parole. Son modèle applicatif cherche à transcrire un signal audio numérisé en un texte écrit. Les difficultés particulières que pose cette forme de transcription (bruitage, difficulté de segmentation, contextualisation) ont conduit à mettre au point un système de reconstruction du signal, inspiré des méthodes de décodage proposées par Shannon. Ici, on considère que le signal bruité transcrit sous sa forme écrite est incomplet et qu'il est possible, par un décodage reposant sur l'exploitation probabiliste d'un modèle de retranscription, de le reconstituer. Le modèle de langage n-grammes sert à élaborer un graphe de probabilité de transitions entre groupes de termes. Ce modèle connaît de nombreuses variantes. Des systèmes tels que ceux préconisés par (Nocera et al., 2002) utilisent des algorithmes d'exploration $A*$ améliorés pour la transcription de signal de parole en temps réel. Pour leur application à la GAT, ils nous intéressent en tant que dispositifs capables de reconstruction de phrases incomplètes permettant d'identifier des mots manquants. Nous avons postulé (Charton et al., 2008) qu'ils sont capables de ré-écrire une phrase artificiellement bruitée (c'est-à-dire dont on aurait supprimé une partie des mots en vue de la soumettre à un processus de reconstruction). La principale difficulté rencontrée lors de la génération statistique de phrases est proche de celle rencontrée dans le problème du décodage de parole ou de la traduction automatique. Cette difficulté réside dans les explosions combinatoires qui découlent des méthodes de ré-assemblage des phrases. On utilise pour résoudre cette difficulté des heuristiques de décodage dont le fondement est le plus souvent l'équation générique du cadre probabiliste bayesien, appliquée à la composition ou à la reconstruction de phrases d'après un modèle de langage :

$$W' = argmax_w P(O|W)P(W) \tag{3.6}$$

où :

$$O = o_1 P(O|W)P(W), ..., o_T \tag{3.7}$$

est la séquence d'observations représentant le signal d'entrée plus ou moins bruité (le signal audio à contenu vocal, les phrases à traduire, des phrases incomplètes, par exemple), vu durant un temps T, et $W = w_1, ..., w_n$ une séquence de mots W pris dans un vocabulaire de taille n, d'après un corpus d'apprentissage. En pratique on gagne à remplacer les mots w par des séquences d'états HMM indépendants du contexte, dont les probabilité sont celles d'apparitions de n-grammes (voir (Katz, 1987) et (Aubert, 2002) pour un tutoriel plus détaillé). Dans ce cas, la séquence de mots reconnus W pour un signal d'entrée O est déterminée par la séquence dont les états sont les plus probables, après exploration des réseaux bayésiens. Dans les applications de traduction automatique comme dans celles de décodage d'un signal de parole, ces algorithmes ont pour seule finalité de conditionner une probabilité de sortie d'après une observation. On n'est donc pas dans un système directement adaptable à une génération de texte. L'*Intention de Communication* a sa logique propre qui consiste en une expression sémantique hiérarchisée et formalisée dont doit découler une phrase exprimant parfaitement l'idée. La phrase générée est une création complète. Dans l'activité de décodage, il n'y a pas création mais validation ou correction d'une phrase dont le sens et la syntaxe sont déjà existants dans le signal de parole. Ces méthodes consistent donc en des corrections et des mises en correspondance qui ne sont en tant que tel pas adaptables à une activité de génération. On peut cependant les impliquer dans des étapes spécifiques de l'architecture *pipeline* d'un système de GAT : en particulier la phase de *Génération de forme de surface* où elles peuvent corriger un un texte généré ou encore le paraphraser. Nous avons notamment étudié cet aspect dans une application de génération exhaustive d'après un sac de mots (Charton et al., 2010a). Nous avons également envisagé qu'il soit possible de reprendre ce modèle pour substituer localement des sections de phrases telles celles qui entourent un connecteur logique (Charton et Torres-Moreno, 2010).

3.2 Éléments statistiques de générateurs de textes

Depuis une dizaine d'années, des auteurs ont envisagé d'utiliser des systèmes statistiques hybridés avec des méthodes de contrôle syntaxique gérées par des grammaires. Les systèmes de GAT reposant sur des méthodes statistiques peuvent être classifiés de la manière suivante :

- **Hybrides à base composants linguistiques**. La plupart des systèmes à base de statistiques associent un composant linguistique avec un système de type statistique.
- **Corpus et apprentissage automatique**. On fait référence ici à tout système reposant sur un apprentissage d'après des données pour produire des phrases (on pourra se reporter à l'exemple de (Corston-Oliver et al., 2002)).
- **À base de modèles n-grammes**. Des méthodes de GAT utilisant des ML pour gérer tout ou partie de la phase de génération de surface.

De nombreuses expériences ont été menées depuis les débuts de la GAT qui utilisent l'une ou l'autre ces techniques, prises isolément ou combinées.

Le système PENMAN avec modèle de langage

Une approche de modèle hybride de génération de texte est présentée dans (Knight et Hatzivassiloglou, 1995). Ce système a été mis en place dans le cadre d'un système de traduction de quotidiens japonais en anglais. L'idée est de simplifier la conception du générateur en déléguant des aspects du choix lexical (sélection de prépositions ou encore contraintes inter-lexicales non compositionnelles) à un composant statistique. Ce composant intervient pour simplifier les règles de génération. Le générateur de phrases, peu sophistiqué, est ainsi autorisé à produire des phrases partiellement erronées en considérant qu'elles pourront être corrigées à posteriori par le composant statistique, à la manière d'un correcteur grammatical d'un traitement de texte par exemple (principe décrit dans 3.1.2, et exploré dans (Charton et al., 2008)).

Knight et Hatzivassiloglou [4] suggèrent que l'un des principes de ce type de composant pourrait être de considérer que pour une entrée I, il soit possible d'utiliser une base de connaissances qui proposerait une phrase A ou B susceptible de représenter I. Un tel composant doit être en mesure de fournir un score permettant de choisir entre A et B, en comparant $I \leftarrow A$ et $I \leftarrow B$. L'une des possibilités pour calculer ce score pourrait être d'utiliser la qualité d'expression de A et de B sans considérer I. Ce point de vue (qui laisse de nombreuses questions ouvertes , notamment celles relatives à la capacité de A et B à restituer le sens de I) est proposé en postulant que le générateur a produit une adaptation de I dans A ou B qui soit correcte. Les auteurs proposent que la qualité d'expression soit évaluée avec un estimateur de probabilité basé sur un modèle de langage bi-gramme et tri-gramme calculé d'après un corpus de 46 millions de mots issu du Wall Street Journal.

L'idée est de rechercher dans un corpus la meilleure *phrase candidate* pour un I donné. En utilisant les formules appliquées aux bi-grammes (formule 3.8) et tri-grammes (formule 3.9), les probabilités conditionnelles sont converties en log-vraisemblance (*log-likelihood*) et utilisées pour estimer la meilleure probabilité $P(S)$ qu'une phrase anglaise corresponde à I selon la propriété de Markov. Une pondération est appliquée selon la longueur de la phrase par l'entremise d'une méthode heuristique (empirique et triviale, puisqu'elle se résume à la fonction $f(l) = 0.5 * l$) pour ne pas discriminer les phrases trop longues que le produit de probabilité désavantage.

$$logP(S) = \sum_i logP(w_i|w_{i-1}) \tag{3.8}$$

$$logP(S) = \sum_i logP(w_i|w_{i-1}, w_{i-2}) \tag{3.9}$$

(Knight et Hatzivassiloglou, 1995) présentent les résultats d'une expérience menée en intégrant leur composant avec le système PENMAN (Bateman, 1990). Une représentation symbolique est générée d'après une phrase en japonais fournie à PENMAN pour produire des propositions de phrases comme dans les 3 exemples de la figure 3.1. Puis

4. Lire (Knight et Hatzivassiloglou, 1995) page 254.

A new company will have in mind that it is establishing it on February.
The new company plans the launching on February.
New companies will have as a goal the launching at February

TABLE *3.1* – *Représentations symboliques fournies par PENMAN.*

Rang	Phrase candidate	score
1	The new company plans to launch it in February.	-13.5682
2	The new company plans the foundation in February.	-13.7551
60	The new companies plan the establishment on February.	-16.3501
401	The new companies will have in mind to establish it at February.	-23.8423

TABLE *3.2* – *Exemple de listes de phrases obtenues par la version de PENMAN modifiée d'après des représentations symboliques.*

la version de PENMAN modifiée est utilisée pour produire 3456 phrases en anglais d'après ces propositions initiales. Le calcul de score produit une liste triée dont un échantillon est donné dans la figure 3.2.

Le système NITROGEN

Cette première étape est suivie de la mise au point du système NITROGEN (Langkilde et Knight, 1998a). NITROGEN utilise une grammaire écrite manuellement dont le rôle est de mettre en relation une représentation sémantique et une séquence de mots en utilisant des contraintes de linéarisation. Dans ce système, une structure sémantique complexe est transformée en graphe et il est fait appel à un modèle de langage bigramme pour choisir au sein du graphe les constituants qui correspondraient le mieux à une *Intention de Communication*. Ce système propose son architecture propre, représentée dans la figure 3.1. Son fonctionnement est décrit dans (Langkilde et Knight, 1998b). Le processus consiste en une première étape de mise en relation d'une entrée avec un graphe de mots. Ce graphe de mots est une représentation compacte de tous les chemins possibles de génération. La mise en relation conduit à proposer plusieurs chemins de parcours du graphe. La représentation symbolique (produite par le module intitulé *générateur symbolique* dans la figure 3.1) est conservée en tant que liste de relations entre les mots et leurs concepts de sens tels que définis dans l'ontologie. La liste est composée de tuples sous la forme indiquée dans 3.10.

$$< mot >< part - of - speech >< rang >< concept > \tag{3.10}$$

qui pourrait avoir la forme de cet exemple :

("eat" VERB 1 | eat , take in |)
("eat" VERB 2 | eat > eat lunchl |)
("take in" VERB 14 | eat , take in |)

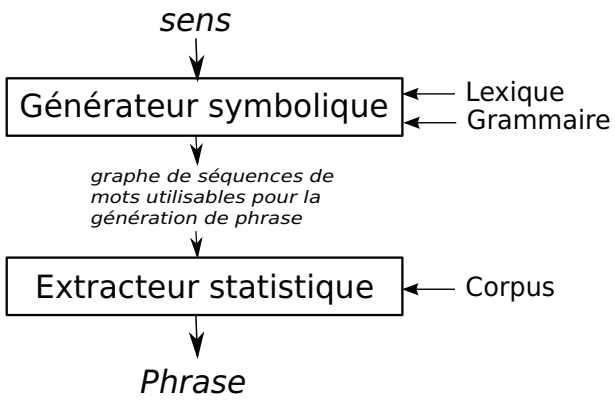

FIGURE 3.1 – *L'architecture du système NITROGEN décrite dans (Langkilde et Knight, 1998a).*

Le champ de rang ordonne les concepts par fréquence de sens identiques pour le mot concerné. Une valeur plus faible signifie donc un sens rencontré moins fréquemment. Un extracteur statistique (voir figure 3.1) est ensuite chargé de calculer le meilleur chemin dans ce graphe par une approximation d'après la formule 3.11 qui évalue que la probabilité qu'une phrase $w_1....w_n$ soit compatible avec l'*Intention de Communication*.

$$P(w_1|START)P(w_2|w_1)...P(w_n|w_{n-1})P(END|w_n) \tag{3.11}$$

Les règles de grammaire de NITROGEN (introduites au niveau du *Générateur Symbolique* dans le système décrit par la figure 3.1) sont organisées pour prendre en compte les relations sémantiques. Un exemple de ces grammaires est donné ci-dessous :

```
Semantic-- :agent, :patient, :domain, :range, :source, :destination,
:spatial-locating, :temporal-locating, :accompanier;
Deep Syntactic-- :obliquel, :oblique2, :oblique3, :adjunct;
Shallow Syntactic-- :subject, :object, :mod, :tense, :quant,
:definiteness, etc.
```

L'*Intention de Communication* représentée par un graphe tel que celui de la figure 3.2 sert de support à la production d'une génération symbolique qui consiste en un second graphe inventoriant tous les chemins possibles. Le graphe qui dérive de la figure 3.2 contient 245 nœuds, 659 arcs, pour un potentiel de 11 664 chemins. Il n'est pas précisé exactement par quel moyen les mots outils tels que *who, in, by, a*, sont intégrés dans les chemins possibles mais on déduit que les règles de production se sont acquittées de cette tâche. Un modèle de langage 2-gramme calculé sur le corpus du *Wall Street Journal* est ensuite utilisé pour rechercher les meilleurs chemins. Les 10 chemins de probabilités les plus élevées sont listés dans la figure 3.3.

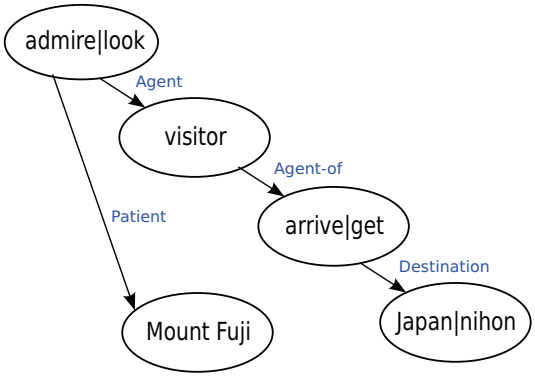

FIGURE 3.2 – *Le graphe de NITROGEN représentant l'Intention de Communication.*

Rang	Phrase
1	Visitors who came in Japan admire Mount Fuji .
2	Visitors who came in Japan admires Mount Fuji .
3	Visitors who arrived in Japan admire Mount Fuji .
7	The visitor who came in Japan admire Mount Fuji .
8	Visitors who came to Japan admires Mount Fuji .

TABLE 3.3 – *Représentations symboliques fournies par NITROGEN.*

3.3 Utilisation de corpus dans la génération de texte

Dans la plupart des systèmes qui dérivent de NITROGEN, on conserve l'idée de produire massivement des phrases via des règles puis de sélectionner la plus appropriée. On retrouvera cette idée dans (Bangalore et Rambow, 2000a). Des méthodes empiriques ont aussi été appliquées pour résoudre avec des approches à base de corpus des tâches connexes de la GAT telles que la sélection lexicale (Bangalore et Rambow, 2000a) l'organisation morphosyntaxiques d'une phrase ou encore la structure de la phrase (Lapata, 2003). On pourra aussi mentionner les travaux de (Kan et McKeown, 2002), de (Chen et al., 2002) et plus récemment de (Marciniak et Strube, 2005). Dans cette dernière proposition, le corpus est utilisé en tant que ressource de connaissance linguistique capable de piloter l'étape de *Génération de surface*. La connaissance issue du corpus est employée pour mettre directement en relation une *Intention de Communication* avec sa forme linguistique. Trois étapes principales suffisent selon l'auteur :

1. Modularisation abstraite du processus de génération en tâches de bas niveaux qui vont rendre possible une réalisation de surface d'après des connaissances issues d'un corpus.
2. Annotation du corpus avec des étiquettes sémantiques et grammaticales.
3. Choix d'une architecture *pipeline* adaptée aux transformations successives de texte dans le contexte applicatif particulier d'un système à base de corpus.

Le corpus utilisé comme ressource par (Marciniak et Strube, 2005) est de petite taille. Il contient 75 modèles de guidage du texte avec un total de 904 unités discursives. Les annotations linguistiques se répartissent en 4 catégories :

1. Étiquetage des unités discursives et mise en correspondance avec des noyaux.
2. Étiquetage des prédicats correspondants aux verbes principaux des noyaux.
3. Étiquetage des arguments syntagmatiques des verbes.
4. Étiquetage des conjonctions et des marqueurs de discours.

Il est précisé que pendant la phase de préparation des données en vue de l'apprentissage, les phrases reçoivent une étiquette morphosyntaxique (POS). Les étiquetages discursifs et de relations sont réalisés avec un système à règles conçues manuellement.

Le système SPoT

SPoT, de (Chen et al., 2002), intègre un composant statistique à base de corpus dans la phase de planification de phrases en suivant deux étapes : dans un premier temps, le *Planificateur de phrases* génère un ensemble de plans pour les phrases possibles [5] d'après une *Intention de Communication* fournie par le *Gestionnaire d'Intention de Communication*. Ces plans sont générés aléatoirement par une construction incrémentale d'après une distribution de probabilité. Dans un second temps, un *Évaluateur de plan de phrases* produit une liste triée des plans selon l'estimation de probabilité.

Le système FERGUS

Le système FERGUS décrit par (Bangalore et Rambow, 2000b) dérive de NITROGEN à quelques détails près (notamment l'adoption d'un modèle de langage 3-gramme). Il consiste en un processus de génération associant un corpus annoté sous forme d'arbre à des grammaires GTAG [6] écrites manuellement. FERGUS est composé de trois modules : le sélectionneur d'arbre, le décomposeur de séquences (*unraveler*) et le sélectionneur linéaire de priorité. Le sélectionneur d'arbre associe un *supertag* à une grammaire par arbre joint (TAG) (Bangalore et Rambow, 2000a). Les plans sont produits par le *Planificateur de phrases* de SPoT et servent à guider le processus de fabrication de la surface de phrases. Le système final est implémenté dans un système de dialogue. L'association de FERGUS et de SPot donne un bon exemple des avantages que procure l'assemblage de modules issus de recherches variées pour mettre au point ou tester des éléments de systèmes de GAT, comme nous l'avions souligné dans la section 2.2. Les concepteurs de SPot soulignent d'ailleurs ce point en conclusion de leur article [7].

5. L'auteur parle de 12-20 phrases possibles.
6. La grammaire d'arbres adjoints, ou grammaire TAG, est un formalisme d'analyse grammaticale reposant sur un système de réécriture d'arbres.
7. *«By integrating SPoT with FERGUS, we have also shown that different statistical NLG components can be made to work well together.».*

3.4 Conclusion

Nous n'avons pas identifié d'architecture entièrement statistique de GAT. Les expériences menées pour intégrer un composant de nature statistique dans une architecture *pipeline* ont la plupart du temps pour finalité de compléter un composant de génération reposant sur une grammaire. On relève que les modèles de langages à base de *n*-grammes, qui ont fait leurs preuves dans de nombreuses applications du TAL, ont aussi fait l'objet d'études en GAT et donnent des résultats encourageants dans le cadre d'une phase de réalisation linguistique finale. On note que ces composants statistiques font parfois appel à des étiqueteurs directement issus des méthodes automatiques. Les étiqueteurs syntaxiques sont les plus fréquents (système de Marciniak vu en 3.3, et NITROGEN vu en 3.2). Plusieurs systèmes exploitent des corpus en tant que ressource de contrôle de la réalisation linguistique. Globalement, ces systèmes qui font intervenir une connaissance issue de corpus ou de modèle de langage sont moins performants que les systèmes à base de règles pour élaborer une forme de surface conforme aux bons usages d'une langue. On mentionnera que les systèmes à base de composants statistiques sont également considérés comme lents du fait de leur complexité et des phénomènes d'explosion combinatoire qu'ils engendrent et font régulièrement l'objet de critiques sur ce point (Knight et Hatzivassiloglou, 1995; Belz, 2005; Langkilde, 2000).

Chapitre 4

Proposition d'architecture à base de *phrases modèles*

Sommaire

Le chapitre 1 nous a conduit à identifier dans la littérature les trois composants dits *consensuels* d'une architecture de système de GAT. Nous avons également pu observer que la hiérarchisation de ces trois composants étaient communément réalisée au cœur d'une architecture de type *pipeline*. Dans le chapitre 2 nous avons vu comment l'architecture dominante de génération de texte exploite des composants linguistiques à base de grammaires pour produire la forme de surface d'une phrase d'après une *Intention de Communication*. Dans le chapitre 3 nous avons montré que plusieurs alternatives sont proposées pour compléter les composants linguistiques à base de grammaire par des approches à base de méthodes statistiques. Nous avons observé que ces méthodes statistiques exploitent le plus souvent des modèles de langage n-grammes, ce qui les rapprochent d'autres familles applicatives du TAL telles que la Traduction Automatique (TA) ou encore le Reconnaissance Automatique de la Parole (RAP). Nous avons précisé que certains systèmes de GAT tentaient d'exploiter des ressources lexicales ou ontologiques apprises au besoin par méthode statistique, pour enrichir le processus de génération. Nous avons souligné qu'aucun système de GAT entièrement statistique n'existait. Nous avons vu que, parfois, les composants grammaticaux des générateurs peuvent se réduire à une forme rudimentaire, par exemple en produisant des phrases

dont la qualité de surface serait dégradée et améliorée à posteriori par approche statistique. Mais globalement aucun système de GAT présenté jusqu'ici ne se dispense d'un composant de production de phrase reposant sur un minimum de règles de productions grammaticales écrites par un expert humain. Par ailleurs, nous avons pu remarquer que les systèmes à base de patrons décrits dans le chapitre 1, bien que plus répandus et anciens, ont un statut à part. On aura remarqué que le potentiel de ces systèmes n'a été que peu exploré, principalement pour la raison que 70% des générateurs décrits implémentent des méthodes de génération à base de règles difficilement compatibles avec les systèmes à base de patrons tels qu'ils sont conçus aujourd'hui.

4.1 Avantages et inconvénients des différentes approches

L'examen des différentes architectures de GAT connues et du rôle que les différentes théories linguistiques jouent dans ces architectures nous inspire les réflexions suivantes sur leur avantages et inconvénients.

Systèmes à base de patrons. Les architectures à base de patrons sont simples à déployer, peuvent être le cas échéant multilingues, mais résolvent mal des tâches de génération complexes qui impliquent des transformations intervenant au cœur de la structure de la phrase (i.e un changement de nature de mot).

Systèmes *pipeline* incluant des composants linguistiques à base de grammaires. Au contraire, les systèmes à base de composants linguistiques à règles de génération grammaticales, insérés dans une architecture *pipeline* à trois niveaux s'adaptent particulièrement bien au changement de nature d'un constituant de la phrase (tel que le temps d'un verbe ou le genre d'un mot), grâce à la souplesse de leur module de réalisation de surface. Le défaut de ces systèmes à base de règles est leur grande proximité avec la langue, qui impose d'implémenter au cœur même du programme de génération des règles de production et de transformation difficilement adaptables dans un autre langage (gestion des temps, de morphologie, adaptation selon les genres, etc.).

Systèmes *pipeline* incluant des composants linguistiques à base de statistiques. Le systèmes statistiques se heurtent à un double problème : les phénomènes d'explosion combinatoires rencontrés dès qu'il est question de produire des phrases d'après un modèle de langage, et le manque de solutions disponibles pour guider la construction de phrases d'après ce même *modèle de langage* (ML). Tout semble montrer que le trop grand éloignement entre la réalité sémantique du choix lexical et la forme du ML bride ces systèmes. Finalement les concepteurs de systèmes statistiques sont contraints d'hybrider leurs applications avec des systèmes à base de règles, ce qui en limite le potentiel puisque la partie statistique du système est restreinte à un traitement correctif mineur de la surface des phrases.

4.2 Proposition

Deux grands courants de pensée se heurtent lorsqu'il s'agit de modéliser les structures d'une langue d'après ses constituants élémentaires. L'approche chomskienne historique [1] (qui prévaut dans les systèmes de GAT à base de règles) décrit la langue comme un système infini et non dénombrable, et non modélisable par des approches statistiques, dans laquelle la syntaxe prime sur le sens.

D'autres auteurs, inspirés par Davidson, considèrent à l'opposé que la structure sémantique d'une phrase prime sur sa syntaxe et peut aider à créer un formalisme universel du langage. La pensée de Davidson sur la théorie sémantique est développée sur la base d'une conception holistique [2] de la compréhension linguistique [3]. Selon lui, proposer une théorie du sens pour un langage revient à développer une théorie qui produirait, pour n'importe quelle phrase existante ou potentielle du langage en question, un théorème qui décrit ce qu'elle signifie. Ainsi, une théorie du sens de l'anglais, qui serait donnée en français, pourrait expliquer que le sens de la phrase en anglais `snow is white`, correspond en français à `la neige est blanche`. Le sens, caractériserait donc la structure profonde de la phrase, devenue abstraite et universelle (pour une espèce parlante donnée) et reléguant la syntaxe au second plan, celui d'un ensemble de mécanismes de production dédiés à la formation de surface de la phrase, destinée à n'être plus qu'un véhicule du sens, une interface de communication. Suivant Davidson, nous avons envisagé un système de GAT qui chercherait pour exprimer une IC une phrase déjà connue, par exemple observée dans un corpus.

4.2.1 Formalisme à base de DRT

Ceci revient à insérer dans une architecture *pipeline* un composant de génération de surface qui utiliserait des modèles de phrases appris sur un corpus et représentées avec un formalisme logique d'inspiration davidsonienne. On aura pas à proprement parler reproduit dans le système de GAT le raisonnement logique qui conduit à la formation d'une phrase au sens davidsonien, mais plutôt simulé ce mécanisme en transformant, par une approche pragmatique, des phrases existantes en patrons réutilisables, décrits par une formule logique. Cette approche est explorée sous l'angle du raisonnement à base de cas (CBR) dont un état de l'art détaillé à été présenté dans (Lamontagne et Lapalme, 2002). L'idée du CBR est d'exploiter une méthodologie très proche de celle que nous proposons non pas appliquée à la production textuelle, mais plutôt à la recherche et à l'extraction d'information depuis des phrases en combinant des formules sémantiques avec des méthodes statistiques. Notre idée est que l'existence de très grand corpus (Wikipédia, projet Gutemberg, pages Web), conjuguée à la disponibilité d'outil de

1. Nous croyons utile de préciser *historique* car au cours des deux dernières décennies, l'approche chomskyenne a été profondément transformée, y compris par son auteur. Ces transformations ne nous semblent pas encore totalement intégrées dans le TAL en général et la GAT en particulier (Fitch et al., 2005; Chomsky, 2005; Hauser et al., 2002).
2. Qui relève du holisme, considérant l'objet comme constituant d'un tout.
3. Lire *Truth and Meaning*, 1967.

traitement automatique des langues (en particulier d'étiquetage et d'analyse lexicale et sémantique), rend aujourd'hui envisageable l'idée que des phrases pré-existantes puissent suffire à répondre au besoin de génération. Par exemple, nous postulons que les 27 millions de phrases de Wikipédia en français ou les 90 millions de phrases de Wikipédia en Anglais pourraient suffire à représenter une proportion importante des *Intentions de communication* contenues dans cette encyclopédie. Nous nous appuyons pour développer cette proposition sur la *Théorie de la représentation du discours (Discourse Representation Theory, DRT)* (Kamp, 1988). La DRT est devenue un nom générique pour englober toute les formes d'interprétations dynamiques du langage naturel. Elle est exploitée dans le cadre de la tâche d'analyse sémantique et fait l'objet d'un intérêt croissant dans ce cadre (Bos, 2008; Jan van Eijck, 2005) notamment en raison de son potentiel de description du contenu sémantique d'une phrase, selon un formalisme très proche de la théorie davidsonienne. Elle fût étudiée en GAT par (Gagnon et Lapalme, 1996a). Considérons dans le cadre de la logique des prédicats un langage L. Ce langage prévoit deux variables :

- u qui est une description sémantique exprimée en *logique de prédicats*
- p qui est une phrase syntaxiquement correcte

et un prédicat :

- S qui définit la relation sémantique entre u et p

$$\forall p \exists p \{S(p, u)\} \tag{4.1}$$

On applique dans L la formule 4.1. Soit pour toute phrase p contenue dans L, il existe une représentation sémantique formelle u. Étant donné les objectifs que nous avons ici, on peut s'interroger sur le rapport entre phrase (au sens linguistique) et proposition (au sens logique).

4.2.2 Représentation sémantique des *phrases modèles*

Ce rapport n'est pas direct : deux phrases peuvent exprimer la même proposition (par exemple dans deux langues différentes, ou encore lorsque l'on utilise des variantes lexicales (comme par exemple *dame/femme* ou *voiture/automobile*). On aura pris soin aussi de préciser que p est obligatoirement syntaxiquement correcte puisqu'un simple assemblage de mots tel que :

```
furieusement incolore hippopotame dorment verte souris
```

ne permet pas de vérifier S. La proposition de 4.1 est donc hypothétique tant qu'aucun moyen ne permet de démontrer qu'il n'existe pas dans L de représentation sémantique non exprimable par une phrase ou de phrase non représentable en logique formelle. Elle nous permet néanmoins de poursuivre la réflexion en l'appliquant cette fois à un corpus de phrases. Considérons l'exemple suivant utilisé par Kamp :

```
Les fermiers qui possèdent un âne le battent.
```

On peut envisager de représenter cette phrase dans le formalisme de la DRT inspiré de la logique formelle par 4.2.

$$((x,y)(\text{fermier(x)}, \text{âne}(y), \text{possède}(x,y))) \Rightarrow m \Rightarrow (()(battent(x,y))) \tag{4.2}$$

Si nous mettons en relation avec chaque phrase d'un corpus, sa représentation sémantique formelle, nous postulons qu'il est possible de réutiliser pour générer un texte, une phrase dont le contenu sémantique correspond à une IC. La méthode que nous proposons pour y parvenir est de rendre toutes les phrases contenues dans un corpus les plus abstraites possibles. Ceci permettra de les réutiliser pour exprimer une IC donnée. Ce qui revient à dire, si nous considérons l'exemple de phrase cible donné dans le chapitre 1 :

```
John possède une grande maison blanche située rue de Paris,
à proximité de l'école primaire.
```

Qui peut se formaliser en IC selon la DRT indiquée en 4.3.

$$DRT = \begin{pmatrix} (x,y,z,b,w) \\ (\text{john(x)}, \text{maison}(y), \text{rue de Paris}(z), \text{blanche}(b), \text{école primaire}(w)) \\ (\text{possède}(x,y))), \text{couleur}(y,b), \text{localisée}(y,z), \text{localisée}(y,w)))) \end{pmatrix} \tag{4.3}$$

Que pour produire IC avec des *phrases modèles*, nous devons définir une méthode de recherche dans un corpus de phrases d'un échantillon pré-existant susceptible de servir de support, comme par exemple :

```
[Personne] possède une [objet] située [adresse1], à proximité
de [adresse2]
```

et ensuite d'appliquer au modèle de phrase extrait une fonction capable de lui faire exprimer IC par transformation. Notre proposition postule donc qu'il est possible d'adapter les techniques de systèmes à base de patrons en les appliquant à des *phrases modèles* apprises depuis un corpus - qui deviennent alors autant de patrons - utilisables par des composants de *Planification de phrase* et de *Génération de forme de surface* d'un système de GAT qui assureraient la fonction de transformation. Nous voyons plusieurs avantages à cette proposition :

- Si elle est applicable à la génération de phrases complexes, elle résout le problème de la génération de texte multilingue en supprimant le composant de génération linguistique et en le remplaçant par une ressource de *phrases modèles* prédéfinies obtenue par apprentissage automatique.
- Si elle est à minima applicable à des phrases simples, elle peut remplacer les *Générateurs de forme de surface* à base de règles, quitte à compléter ces derniers

par un agrégateur chargé de transformer des phrases simples en phrases complexes. On perdrait alors l'avantage de l'apprentissage complètement automatique et indépendant de la langue - puisqu'il faudrait écrire un agrégateur pour chaque version linguistique du générateur - mais on diminuerait tout de même substantiellement le coût d'écriture d'un système de GAT.
– Elle résout le problème de l'adaptation au domaine du composant de génération linguistique puisqu'il suffit d'apprendre les *phrases modèles* sur un corpus d'un domaine spécifique pour augmenter la probabilité qu'à une IC donnée corresponde un modèle de phrase.

Un tel système donne une importance accrue à la phase d'apprentissage, qui devient une étape constitutive et indissociable du système de GAT, là où les systèmes existants sont souvent hautement contraints par leurs composants linguistiques. Il implique par ailleurs de mobiliser toutes les ressources du TAL pour rendre les phrases modélisées adaptables, et donc les plus abstraites possibles. Cette possibilité est aujourd'hui permise par les progrès importants réalisés dans des applications d'étiquetage (*Part Of Speech*, analyse des dépendances, entités nommées) au cours des dix dernières années. Notons que le cadre formel de la DRT prévoit bien plus que la seule relation logique entre constituants. Il formalise notamment les expressions temporelles. Cette caractéristique explorée par ailleurs (Gagnon et Lapalme, 1996a) ne sera pas étudiée ici. Nous nous en tiendrons strictement à l'étude de la capacité pour un système à passer d'une IC à une phrase, via un formalisme proche de la DRT. Mais notre système - dans une seconde phase de recherche - pourrait s'hybrider avec un composant de gestion des expressions temporelles tel que celui décrit dans (Gagnon et Lapalme, 1996b) dont le formalisme des IC est très proche du nôtre.

4.2.3 Exemple

Pour clarifier notre proposition, nous l'illustrons avec un exemple détaillé. Considérons que les ressources linguistiques qui se substituent aux règles de production grammaticale dans le *Planificateur de phrases* sont des *phrases modèles* apprises par une méthode automatique, en s'aidant de techniques d'étiquetage statistiques. Il existe une grande proximité entre cette proposition et le système de (Kosseim et al., 2001) qui consistait à créer des patrons phrastiques d'après des courriels.

Pers=Romain Gary	
Pers.pseudonyme	Émile Ajar
Pers.fonction	Écrivain
Pers.bornplace	Vilnius
Pers.deathplace	Paris
Pers.borndate	8 mai 1914
Pers.deathdate	2 décembre 1980
Pers.nationalité	Français

TABLE 4.1 – *Exemple d'entrée d'une base de donnée utilisée pour générer un texte.*

Pour avancer dans la description de la méthode envisagée, considérons une base de données de nature bibliographique, qui décrirait des personnes et certains de leurs attributs distinctifs. L'une des entrées de cette base de donnée pourrait être celle représentée dans la table 4.1. Elle peut être formalisée par la DRT 4.4.

$$DRT = \left(\begin{array}{l} (a,b,c,d,e,f,g) \\ \\ (\text{Romain Gary(a)}, \text{Émile Ajar}(b), \text{écrivain}(c), \text{Vilnius}(d), \text{8 mai 1914}(d), \\ \text{français}(e), \text{Paris}(f), \text{2 décembre 2980}(g)) \\ \\ (\text{Pseudonyme}(a,b), \text{Fonction}(a,c), \text{Bornplace}(a,d), \text{Borndate}(a,d), \\ \text{Bornplace}(a,d), \text{Deathdate}(a,g)\text{Deathplace}(a,f)) \end{array} \right) \tag{4.4}$$

Nous pouvons envisager qu'il existe dans un grand corpus de bibliographie de très nombreuses phrases déjà écrites qui peuvent **véhiculer** l'*Intention de Communication* contenue dans la table 4.1 et formalisée par 4.4. Un bref échantillon collecté manuellement dans les fiches encyclopédiques de Wikipédia qui décrivent des écrivains français nous donne les propositions suivantes :

1. `Honoré de Balzac, né Honoré Balzac, à Tours le 20 mai 1799 (1er prairial an VII) et mort à Paris le 18 août 1850`
2. `Alphonse de Lamartine, de son nom complet Alphonse Marie Louis de Prat de Lamartine, né à Mâcon le 21 octobre 1790 et mort à Paris le 28 février 1869.`
3. `Gérard de Nerval, pseudonyme de Gérard Labrunie, né à Paris le 22 mai 1808 et mort à Paris le 26 janvier 1855, était un poète français.`
4. `Jean-Paul Sartre (Jean-Paul Charles Aymard Sartre), né le 21 juin 1905 à Paris et mort le 15 avril 1980 à Paris, est un philosophe et écrivain français.`

Si certaines de ces propositions sont relativement éloignées de IC on note que d'autres lui sont adaptables en l'état. Ainsi la proposition 3 :

- `Gérard de Nerval, pseudonyme de Gérard Labrunie, né à Paris le 22 mai 1808 et mort à Paris le 26 janvier 1855, était un poète français.`

Peut être rendue plus abstraite en remplaçant ses entités nommées par des étiquettes :

- `[PERS], pseudonyme de [PERS], né à [LOC] le [DATE] et mort à [LOC] le [DATE],était un [FONC] [ORG.GSP].`

On peut ensuite par un processus d'association, relier les entités étiquetées dans le texte à leur correspondantes dans l'*Intention de Communication* :

- **[Pers.pseudonyme]**, pseudonyme de **[Pers]**, né à **[Pers.bornplace]** le **[Pers.borndate]** et mort à **[Pers.deathplace]** le **[Pers.deathdate]**, était un **[Pers.fonc] [Pers.nationalité]**.

Dans notre proposition de système, il serait finalement possible de transformer le modèle de phrase ci-dessus pour qu'il reçoive le contenu de 4.1 :

- **Émile Ajar**, pseudonyme de **Romain Gary**, né à **Vilnius** le **8 mai 1914** et mort à **Paris** le **2 décembre 1980**, était un **écrivain français**.

Pour que notre modèle de GAT fonctionne, nous devons donc d'abord extraire les éléments du texte et les classer dans une perspective de réutilisation (c'est-à-dire de localisation d'un emplacement dans la phrase modèle pour insérer un élément approprié de IC). La tâche de reconnaissance et de regroupement d'entités nommées coréférentes (Belz et al., 2009a) du *Generation Challenge* correspond très exactement à ce besoin, qu'elle pose comme problème. Il nous faudra ensuite bâtir d'après les éléments localisés une représentation logique de la structure sémantique de la phrase selon un formalisme proche de la DRT. Cette tâche est similaire à celle soumise notamment dans les *Shared Task* CoNLL 2008 et 2009 (Surdeanu et al., 2008). Il existe donc des techniques de TAL pour traiter ces aspects difficiles de notre proposition.

4.3 Système de Génération Automatique de Texte à base de *Corpus de Phrases Modèles*

Le système que nous proposons est divisé en deux parties, l'une, intitulée NLGbAse dédiée à l'acquisition de connaissances et à leur structuration, et l'autre à la génération de phrases, qui exploite les connaissances acquises lors de l'apprentissage. Le module de génération de phrase, intitulé NLGEN, repose sur une architecture *pipeline* assurant une transformation par étape d'une IC en une ou plusieurs phrases (voir figure 4.1). Les composants principaux de l'architecture *pipeline* dédiés à la génération de phrases sont donc très classiquement :

1. *Gestionnaire de l'Intention de Communication* (Text Planner) ;
2. *Planificateur de phrase* (Sentence Planner) ;
3. *Réalisateur de phrase* (Sentence Realizer).

Mais contrairement à un système de GAT implémentant un *Générateur de formes de surface* à base de grammaire, les deux dernières étapes de notre générateur exploitent les ressources apprises par approche statistique depuis le corpus pour transformer l'*Intention de Communication* (le «que dire») en phrase (le «comment le dire»).

Les ressources apprises sont des phrases, finement étiquetées, incluses dans un *Corpus de Phrases Modèles (CPM)*. Ceci implique que les composants dédiés à la génération ont besoin de mettre en œuvre méthodes inhabituelles dans un système de GAT pour

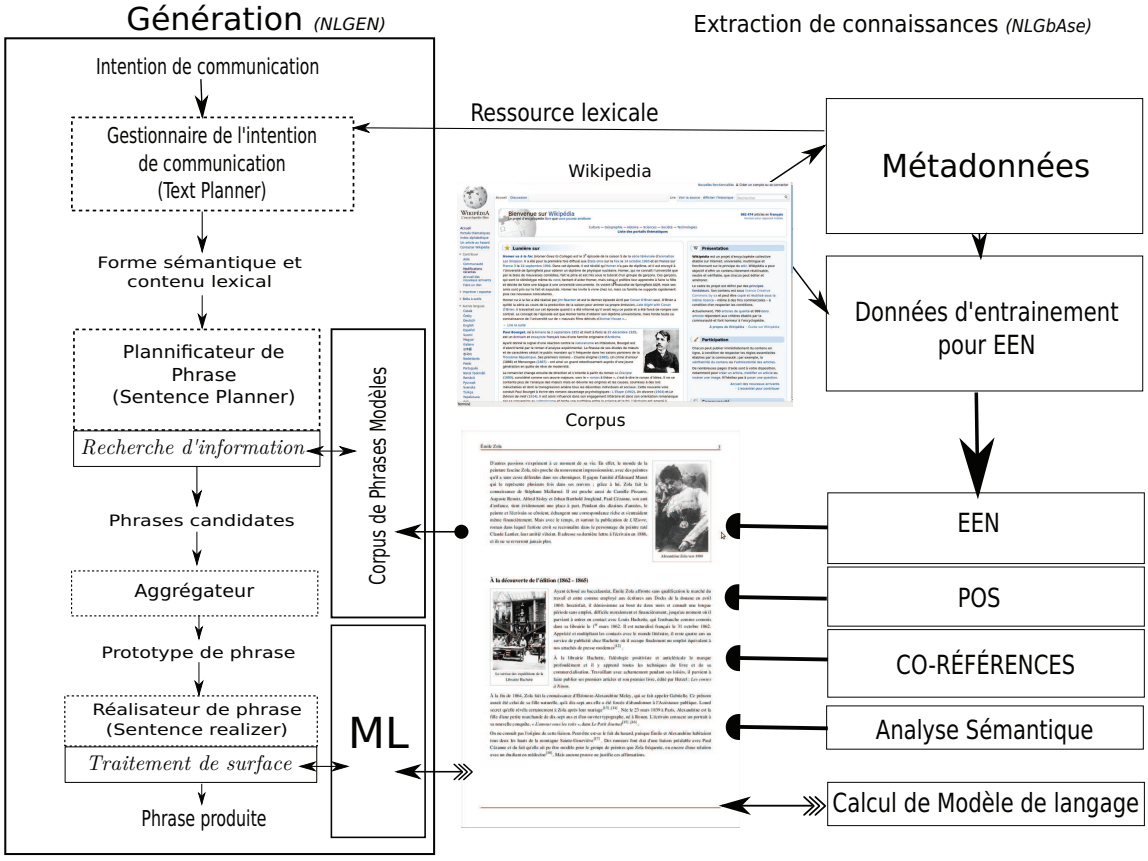

FIGURE 4.1 – *L'architecture complète du système proposé. Un module d'extraction de connaissances (NLGbAse) est utilisé pour produire les ressources et les éléments linguistiques qui seront utilisés par le second module (NLGEN), dédié à la génération de texte.*

fonctionner (voir 4.1). La production grammaticale est par exemple totalement inutile puisque modélisée automatiquement lors de l'apprentissage. En revanche, la récupération d'une *Phrase Modèle (PM)* implique une technique de recherche d'information.

4.3.1 Description du système de génération de texte proposé

Le module d'apprentissage **NLGbAse** est composé d'un ensemble d'applications qui par étapes successives assurent la construction de ressources qui permettront de construire un *Corpus de Phrases Modèles (CPM)* rendues abstraites par un processus d'étiquetage à plusieurs niveaux (EN, POS, lexiques, anaphores et co-références) puis formalisées lors d'un processus d'analyse sémantique. NLGbAse est conçu dans une perspective de génération multilingue, et peut être entraîné pour modéliser plusieurs langues. NLGbAse inclut également un ensemble de composants capables d'extraire des ressources lexicales d'après des contenus encyclopédiques. Ces ressources lexicales sont par exemples des formes de surfaces utilisables pour écrire une même entité textuelle (*International Business Machine* par exemple, peut aussi être écrit *Big Blue* ou

IBM). Elles sont notamment utilisées pour lexicaliser les IC.

NLGEN est le module de GAT. Il exploite les ressources produites par NLGbAse pour générer des phrases d'après une IC. Ce module inclut des composants de formalisation d'une IC, des algorithmes de recherche susceptibles de localiser dans le *Corpus de Phrases Modèles (CPM)* les phrases les mieux adaptées pour exprimer IC. NLGEN assure ensuite la transformation et la correction éventuelle des phrases extraites du *Corpus de Phrases Modèles (CPM)* en phrases compatibles avec IC. Ceci implique la présence de composants complémentaires tels qu'un agrégateur (qui peut assembler plusieurs phrases de CPM pour les rendre compatibles avec IC), un composant de lexicalisation (qui peut enrichir les IC de synonymes afin d'améliorer les performances du composant de RI), ou un *modèle de langage (ML)* associé à un décodeur pour corriger la surface des phrases générées.

Système d'apprentissage NLGbAse

Les ressources et composants du module NLGbAse sont conçus pour fonctionner indépendamment de la langue. Dans le cadre de ce travail, le module NLGbAse est entraîné en anglais, français et espagnol.

- **Construction de ressources lexicales**. Les ressources lexicales décrivent des entités nommées (principalement des noms propres) des mots (verbes, noms communs) et des locutions. La ressource d'entités nommées est construite à partir de Wikipédia. Elles consiste en un ensemble de *métadonnées* représentant tous les concepts contenus dans Wikipédia. Ces concepts décrivent des personnes, des lieux, des organisations, mais aussi des locutions nominales ou des mots caractéristiques (tels que par exemple `Avion à réaction` ou `Haut-Fourneau`). Cette ressource est notamment utilisée pour gérer des synonymes absents d'index terminologiques tel que Wordnet. Elle est également exploitée dans le système d'étiquetage en tant que ressource d'apprentissage.
- **Composant d'étiquetage**. Le composant d'étiquetage de phrase permet de modéliser toutes les phrases du corpus d'apprentissage qui serviront en tant que *modèle de phrase* pour la génération. Ce système d'étiquetage est en mesure de décomposer une phrase en entités nommées, étiquettes morphosyntaxiques, syntagmes.
- **Composant d'analyse sémantique**. Le composant d'analyse sémantique est chargé de transformer une phrase étiquetée en formule de prédicat, en vue de la comparaison des IC avec les phrases contenues dans le CPM.
- **Composant de détection de co-références**. Le composant de détection de co-références est utilisé pour détecter les co-références des entités nommées de nature synonymiques ou pronominales (il, it, they, them, etc).
- **Production du corpus de phrases étiquetées**. Le composant de production de phrases étiquetées (les *phrases modèles*) prend en entrée un corpus d'apprentissage et produit un modèle d'après toutes les phrases qu'il contient. Chaque phrase est finement étiquetée et rendue abstraite en vue de sa réutilisation en tant que *phrases modèle*. Ce composant peut extraire d'après des phrases complexes, des *proto-phrases* de type *Sujet, Verbe, Complément* en vue de leur agrégation.

Système de Génération Automatique de Texte NLGEN

Le générateur de texte NLGEN peut être décomposé en une architecture *pipeline* à trois étapes. En premier lieu, l'*Intention de Communication* est structurée, puis transmise à un *Planificateur de phrase*. La particularité du *Planificateur de phrase* est de reposer sur un algorithme de *Recherche d'information (RI)* dont le rôle est de localiser et d'identifier dans le *Corpus de Phrases Modèles (CPM)* celle qui correspond le mieux à *l'Intention de Communication (IC)* en utilisant une combinaison d'un algorithme vectoriel (recherche par similarité cosinus), et d'un algorithme de comparaison de formules de logique formelle (comparaison de prédicats). Cet extracteur diffère de celui proposé par (Langkilde et Knight, 1998a) pour NITROGEN (voir 3.2) au sens ou il ne se contente pas d'essayer d'appliquer une structure de phrase à un prototype de phrase déjà planifiée. Dans le cas de NITROGEN, la phrase est produite par un *Planificateur de phrase* à base de règles de grammaires puis corrigée. Dans notre système, la phrase est extraite pré-construite du *Corpus de Phrases Modèles (CPM)* produit pendant la phase d'apprentissage. Les *phrases modèles* retenues pour exprimer IC sont finalement transmises au *Réalisateur de phrase* qui va appliquer un traitement de surface proche de celui proposé par le modèle de décodage (adopté également par NITROGEN, et qui est standardisé dans les systèmes de dialogue tel que décrit dans 3.1.2). L'architecture *pipeline* retenue, et les fonctions qu'elle prévoit, est détaillée ci-dessous [4].

- *Gestionnaire de l'Intention de Communication (Text Planner)* : ce composant doit utiliser les ressources lexicales et en particulier les *métadonnées* pour enrichir le champ lexical de l'*Intention de Communication* (un nom de lieu par exemple, sera associé à un réseau lexical de tous les noms similaires - ex Canyon, Vallée, Défilé). Ceci permet au *Gestionnaire de L'intention de communication* de fournir en entrée du *Planificateur de phrase* des éléments riches qui aideront le processus de RI à extraire les phrases les plus adaptées contenues dans le *Corpus de Phrases Modèles* (par exemple, une phrase `je déguste une banane` peut être transformée pour exprimer l'intention `manger;pomme`). L'extension du champ lexical consistera à fournir au *Planificateur de phrase* l'*Intention de Communication* sous une forme complétée par des hyperonimes.
- *Planificateur de phrase (Sentence Planner)* : ce composant est un extracteur fonctionnant avec des méthodes de RI qui vont identifier dans le *Corpus de Phrases Modèles (CPM)* une ou plusieurs propositions, que nous intitulons les *phrases candidates (PC)*, correspondant à la représentation conceptuelle proposée par le *Gestionnaire de l'Intention de Communication*.
- *Aggrégateur* : l'aggrégateur est chargé de proposer une solution alternative au *Planificateur de phrase* lorsque ce dernier n'a pas été en mesure d'identifier dans le *Corpus de Phrases Modèles* au moins une *phrase candidate* susceptible de représenter *l'Intention de Communication (IC)*. Il décompose IC en plusieurs prédicats de type *Sujet, Verbe, Complément* et tente d'identifier les proto-phrases correspondantes pour les assembler finalement dans une nouvelle *phrase candidate*.

4. Pour aider la compréhension du lecteur, nous indiquons entre parenthèses le nom anglais le plus couramment admis dans la littérature pour un module de fonction similaire, tel que décrit dans le chapitre 1.

– *Réalisateur de phrase (Sentence Realizer)* : ce composant reçoit en entrée la *phrase Candidate* et doit assurer deux tâches : gérer le remplacement des formes abstraites contenues dans la PC sélectionnée d'après les éléments fournies par le *Gestionnaire de l'Intention de Communication* (par exemple, l'étiquette `[personne]` de la phrase `[personne] aime manger` par un nom de personne ou une expression telle que `[Il/Elle] aime manger`). Sélectionner le chemin de construction de phrase la plus appropriée dans le *modèle de langage (ML)* et éventuellement appliquer une transformations de surface à la phrase en cours de génération. Il utilisera pour cela un réalisateur de surface proche de celui proposé par (Langkilde et Knight, 1998b).

4.4 Conclusion

Nous avons décrit une proposition d'architecture de génération automatique de texte entièrement statistique à base d'apprentissage de *phrases modèles*, réalisé sur un corpus de grande taille. L'originalité de cette proposition est qu'elle envisage qu'il soit possible de remplacer entièrement un élément de réalisation linguistique et de génération de forme de surface à base de règles par un module de *recherche d'information* appliqué à une ressource de *phrases modèles* obtenue par apprentissage statistique. Notre proposition - bien que très éloignée des propositions dominantes de composants de réalisation de surface - s'inscrit néanmoins dans une architecture classique de GAT de type *pipeline*. Ce choix est fait dans la perspective de rendre le composant de *planification de phrase* intégrable à un système existant, ou susceptible de fonctionner en complément d'un *planificateur de phrase* à base de règles. Le principal avantage que nous voyons dans cette proposition est qu'elle permet d'envisager de produire des systèmes de GAT par apprentissage automatique en diminuant les contraintes liées au domaine sémantique ou à la langue. Il suffit de ré-entraîner le module proposé avec de nouveaux corpus pour le reconfigurer en vue d'une application spécifique à un nouveau domaine ou à une nouvelle langue.

Deuxième partie

Extraction de connaissances

Chapitre 5

Composants d'apprentissage automatique d'un système de GAT à base de corpus

Sommaire

Dans cette partie, nous décrivons les modules de NLGbAse utilisés pour produire les ressources qui seront utilisées par le système de génération NLGEN. Nous avons expliqué dans le chapitre 4 que pour générer du texte, nous utiliserons une phrase existante dont nous aurons rendu les contenus abstraits. Cette phrase existante sera localisée d'après son modèle sémantique formalisé selon les méthodes de la *DRT*. Le formalisme de la *DRT* implique l'utilisation - en plus d'une représentation en logique formelle - de contenus lexicaux qui permettront d'identifier les phrases. Imaginons par exemple que nous souhaitions exprimer par une phrase l'idée {*Jean ;Conduire ;Voiture*}. Nous devons identifier une *phrase modèle (PM)* dont le contenu sémantique est *Sujet* $\rightarrow$ *Conduire* $\rightarrow$ *Objet* mais aussi dont le contenu lexical est compatible avec *Sujet :PERSONNE* $\rightarrow$ *VERBE :CONCEPT(Conduire)* $\rightarrow$ *OBJET :VEHICULE*. Nous devrons aussi identifier des informations morphosyntaxiques qui indiquent les temps des verbes puisque notre générateur repose entièrement sur les formes sémantiques contenues dans le corpus et n'intègre aucune règle grammaticale.

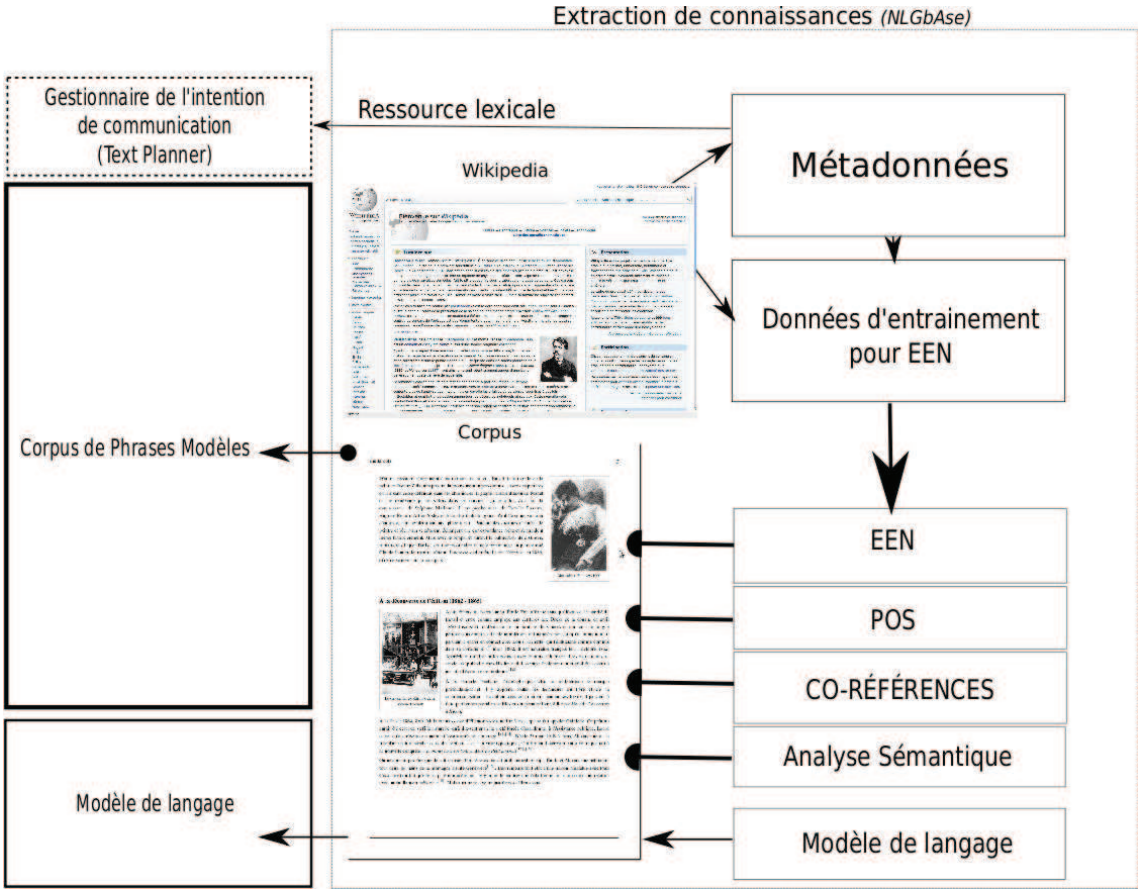

FIGURE 5.1 – *Extraction de connaissances dans l'architecture du système de GAT proposé.*

Ceci implique que nous soyons en mesure d'étiqueter les phrases qui serviront de modèles selon leurs formes sémantiques, leurs contenus lexicaux, et leurs structures morphosyntaxiques (voir la figure 5.2). Il nous faudra aussi résoudre par anticipation certaines ambiguïtés, en identifiant par exemple les anaphores ou les co-références, qui conditionneront la ré-écriture de la phrase selon les éléments d'une *Intention de Communication (IC)*. Dans ce chapitre, nous passons en revue les différents systèmes d'étiquetage et d'extraction d'information disponibles, et les mettons en perspective avec nos propres besoins, en fonction de leur maturité.

5.1 L'étiquetage morphosyntaxique

L'étiquetage morphosyntaxique consiste à assigner à chaque mot d'un texte sa catégorie morphosyntaxique (nom, verbe, adjectif, etc.). L'intérêt de l'étiquetage par POS est qu'il rend possible l'analyse d'un corpus du point de vue linguistique et non plus purement statistique. A partir d'un texte étiqueté, il est en effet possible de caractériser ou de rechercher certaines structures syntaxiques, c'est-à-dire, être capable de s'ab-

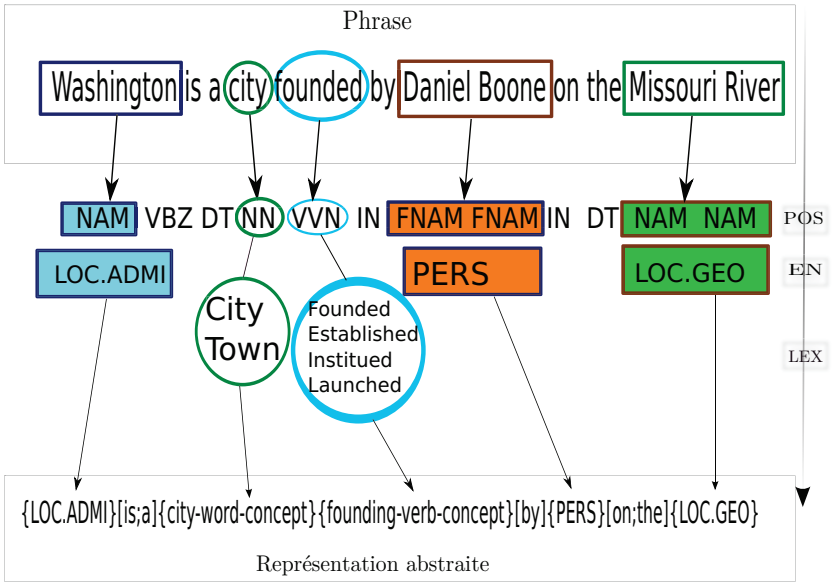

FIGURE 5.2 – *Les différents niveaux d'étiquetage possibles.*

straire du mot pour passer à sa catégorie linguistique. La taille d'un jeu d'étiquettes de POS varie en fonction de la finesse des informations linguistiques recherchée. Elle varie de quelques dizaines à une centaine de classes. Cette taille dépend à la fois de la langue traitée (certaines langues dont la morphologie est très riche nécessitent plus d'étiquettes que d'autres). Par exemple le jeu d'étiquettes utilisé dans le corpus annoté PennTreebank [1] n'en compte que 36, celui de la langue française adopté par TreeTagger 15, avec 33 sous-classes. L'étiqueteur morphosyntaxique va généralement appliquer 3 fonctions pour prendre une décision d'attribution de classe à un mot :

1. il segmente le flux de caractères en mots ;
2. il procède à l'étiquetage a priori (hors-contexte) des mots au moyen d'informations lexicales ;
3. il procède à la sélection en fonction du contexte du mot de l'étiquette la plus pertinente parmi celles identifiées par l'étiquetage a priori.

Les méthodes utilisées sont très variées. Celles à base de règles sans apprentissage ont été les premières employées pour construire des étiqueteurs. Dans ces systèmes, les règles de co-occurrences des mots et des étiquettes, qui définissent l'appartenance d'une occurrence à une classe de mots, sont fournies par un linguiste. Ces méthodes ont été supplantées par des méthodes à base d'apprentissage à partir d'un corpus annoté à la main. Elles utilisent les modèles de Markov cachés (Nasr et al., 2004) les arbres de décision (Béchet et al., 2000; Schmid, 1994). Certaines méthodes à apprentissage - plus anecdotiques - se contentent même d'un corpus dépourvu d'annotations morphosyntaxiques comme par exemple, les méthodes à base de neurones artificiels, de combinaison de systèmes ou encore d'algorithmes génétiques. Certains étiqueteurs peuvent

1. Voir http://www.cis.upenn.edu/~treebank/.

considérer des niveaux d'abstraction plus élevés, incluant jusqu'à la nature sémantique des mots : c'est le cas par exemple de LIA-TAG [2] qui intègre dans sa version 1.1 des étiquettes particulières pour les noms de personnes (XPERS), les noms de lieux (XLOC). Ce niveau supérieur généralement dévolu aux *étiqueteurs d'entité nommées* (EEN) est adopté pour améliorer *à posteriori* les performances d'un système d'EEN à base d'apprentissage statistique (détail en 5.2). D'autres particularités pourront être rencontrées : la version française de TreeTagger étiquette le temps des verbes, ce que ne fait pas LIA-TAG. La version espagnole de Treetagger étiquette les noms de mois, les quantifieurs, les unités de mesure, ce qui est inhabituel mais peut être très utile pour des applications spécifiques.

De manière générale, certains aspects de la tâche d'étiquetage étant très peu ambigus, il est souvent facile d'ajouter un post-traitement qui complète ou adapte un jeu de classes : la détection des noms de mois, de jours, les valeurs numériques écrites en texte sont des exemples de la souplesse et de l'adaptabilité du processus d'étiquetage. Cette faculté est très importante dans notre cadre applicatif : un nom de mois rendu abstrait par une étiquette dédiée, par exemple, autorise le réemploi de la phrase pour n'importe quel mois de l'année.

5.1.1 Maturité des technologies

L'étiquetage morphosyntaxique est une technologie mature, mise en œuvre dans des applications largement diffusées. Il existe actuellement un large éventail de logiciels, libres ou commerciaux, d'étiquetage morphosyntaxique opérationnels pour les langues les plus fréquemment rencontrées (langues romanes, saxones, ou encore en chinois, arabe, ou russe). Les performances revendiquées par les systèmes sont de l'ordre de 95% [3] pour les étiqueteurs français (se reporter à (Paroubek, 2009) pour une évaluation exhaustive récente). Ce niveau de performance, en apparence élevé, souvent affiché par les étiqueteurs morphosyntaxiques et considéré comme trompeur, car il est attribuable, pour une bonne part, au nombre prépondérant de mots non ambigus présents dans la langue naturelle. En pratique, il n'en reste pas moins que la technologie est fiable et peut être déployée en contexte applicatif. Nous avons intégré ce type d'étiqueteur dans notre système en le complétant par un ensemble de règles de détection de classes spécifiques appliquées aux mots substituables (noms de mois, de jours, valeurs numériques, prénoms).

5.2 Étiquetage par entités nommées

La tâche d'*étiquetage par entités nommées* (EEN) est un processus lors duquel chaque mot d'une phrase correspondant à une *entité nommée (EN)* (généralement un nom propre et par extension des dates ou des quantités) reçoit une étiquette de classe. Cette

2. Téléchargeable sur http://lia.univ-avignon.fr/fileadmin/documents/Users/Intranet/chercheurs/bechet/download_fred.html.
3. Voir les campagnes GRACE http://www.limsi.fr/TLP/grace/.

classe correspond à un arbre taxonomique dans la complexité et la nature sémantique peuvent varier (Charton et Torres-Moreno, 2009). Il n'existe pas à proprement parler de standard taxonomique pour les étiquettes. Néanmoins, on observe que les propositions de classes retenues pour les systèmes d'EEN génériques sont généralement issues d'un tronc commun de définitions conceptuelles, proposées lors des campagnes d'évaluation. Dans les systèmes anglophones, ce sont les classes des campagnes MUC [4], ACE (Doddington et al., 2004) ou CoNLL (Tjong et Meulder, 2003) qui sont choisies. Dans le contexte francophone, l'unique disponibilité des corpus de la tâche d'étiquetage de la campagne ESTER 2 (Galliano et al., 2009) a fait des règles taxonomiques proposées par ce groupe un standard de-facto. La tâche d'EEN s'étend à la reconnaissance de locutions nominales (au sens de suite de mots, figée par l'usage, pouvant être substituée à un nom) en regroupant plusieurs mots étiquetés (comme par exemple dans le cas de *Paris* qui est une entité de type ville tout comme *Ville Lumière*, idem pour un produit ou un véhicule tel l'acronyme *TGV* qui décrit le même produit que la locution nominale *Train à Grande Vitesse*).

De nombreuses méthodes ont été proposées pour extraire les EN d'un texte : de la plus simple à base règles, aux plus complexes recourant à l'apprentissage automatique sur corpus étiquetés. La plupart des approches de reconnaissance automatique sont basées sur la théorie des probabilités et peuvent être caractérisées par une méthode générative ou discriminante selon que la distribution de probabilités de la caractéristique à reconnaître est modélisée ou non. Cette différence joue un rôle important dans la tâche d'EEN. En effet, un classifieur discriminant est théoriquement plus précis mais moins capable d'inférer qu'un classifieur génératif et donc moins adaptable aux innombrables possibilités de représentations d'une EEN.

Les deux approches principales se retrouvent dans la tâche d'EEN : les méthodes génératives, comme par exemple celle reposant sur des modèles de Markov cachés (HMM) (Bikel et al., 1999), et les méthodes discriminantes telles que SVM, Maximum Entropie (MaxEnt) (Borthwick et al., 1998). Les Champs Conditionnels Aléatoires (CRF) (McCallum et Li, 2003; Lafferty et al., 2001) occupent une place à part car ils combinent une nature générative et discriminante. Comme les modèles discriminants, ils prennent en compte lors de la construction du modèle de nombreuses observations issues du corpus d'apprentissage et les corrèlent entre elles lors de l'entrainement. Mais à l'instar des modèles génératifs, les CRF probabilisent les décisions en fonction de la position des séquences d'apprentissage. Ce mode de fonctionnement hybride (il favorise l'inférence, c'est-à-dire la reconnaissance par un classifieur CRF d'une entité qu'il n'a jamais observé dans le corpus d'apprentissage, mais aussi la précision en utilisant les données d'apprentissage pour discriminer) explique que de nombreuses études (Raymond et Riccardi, 2007) ont montrées que les systèmes à base de CRF étaient plus performants que ceux à base de HMM, de SVM ou de MaXent pour la tâche d'EEN.

4. Voir http://www-nlpir.nist.gov/related_projects/muc/proceedings/ne_task.html.

5.2.1 Approches à base de CRF

Un CRF peut être défini par un graphe de dépendance G et un jeu de paramètres f_k associés à un poids λ_k. La probabilité conditionnelle d'une annotation y d'après une observation x (dans le cadre applicatif d'une application d'étiquetage) est donnée par [5] la formule 5.1, où $Z(x)$ de 5.1 est un facteur de normalisation détaillé dans 5.2.

$$p(y|x) = \frac{1}{Z(x)} exp\left(\sum_{c \in C}\sum_{k} \lambda_k f_k(y_c, x, c)\right) \tag{5.1}$$

$$Z(x) = \sum_{y} exp\left(\sum_{c \in C}\sum_{k} \lambda_k f_k(y_c, x, c)\right) \tag{5.2}$$

Les paramètres définis *à priori* par le corpus d'apprentissage sont modélisés par les fonctions f_k. Dans la plupart des implémentations de CRF, les paramètres sont encodés sous la forme de suite de fonctions binaires qui retournent 1 en cas de correspondance et 0 en cas de différence. Pour l'exemple suivant :

```
He          PRP  B-NP
reckons     VBZ  B-VP
the         DT   B-NP << position courante
current     JJ   I-NP
account     NN   I-NP
```

Pour lequel seraient élaborées des fonctions correspondants à un graphe G prenant en compte la ligne courante et une ligne précédente et suivante, on aurait les fonctions binaires :

```
func1 = if (output = B-NP and feature="U00:DT") return 1 else return 0
func2 = if (output = B-VP and feature="U-01:VBZ") return 1 else return 0
func2 = if (output = I-NP and feature="U01:JJ") return 1 else return 0
```

Le poids λ_k est associé à chaque paramètre du modèle. Il correspond à la probabilité conditionnelle de y_c (c'est-à-dire la probabilité d'une étiquette d'après la position courante) de la clique c (qui correspond au modèle défini par le graphe G sur le paramètre courant) sachant x. Dans un modèle CRF, la quantité de fonctions générées sera égale au produit du nombre de classes de sorties (par exemples les étiquettes d'EN) par le nombre de chaînes uniques obtenues sur un corpus d'après le graphe G. Le classifieur CRF le plus répandu est CRF++, mais de nombreuses propositions *open source* et diffusées librement existent (CRF Suite, FlexCRFs, etc) [6]. On remarque qu'au delà de la méthode de classification retenue, le choix et la sélection des paramètres utilisés pour l'apprentissage sont particulièrement importants (Ratinov et Roth, 2009). Cette bonne sélection des paramètres doit être par ailleurs associée à une bonne définition du graphe G pour obtenir un système optimal.

5. Pour une description complète du modèle, se reporter à (Lafferty et al., 2001).

6. Une étude comparative des performances des divers classifieurs CRF est disponible sur http://www.chokkan.org/software/crfsuite/benchmark.html.

5.2.2 Maturité des technologies

L'EEN est un domaine parvenu à maturité. La dernière campagne d'évaluation scientifique proposée sur le sujet est la campagne francophone d'ESTER 2. Cette campagne a comblé le manque de corpus d'évaluation et de métriques standardisées pour cette tâche en langue française. Toutes les autres campagnes (ACE, CoNLL) ont été interrompues à la fin de la première décennie 2000. Elles sont remplacées par des évaluations sur l'étiquetage de nature sémantique (KBP, WEPS). La campagne qui devrait suivre ESTER 2 envisage elle aussi de remplacer la tâche d'EEN par une tâche de nature sémantique [7].

Dans le cadre applicatif que nous proposons, l'EEN est stratégique car il doit nous permettre de rendre aussi abstraites que possible (et donc ré-utilisables) les *phrases modèles* apprises depuis un corpus, en remplaçant dans une phrase un nom de personne par une étiquette *personne*, par exemple, ou celui d'une localité par une étiquette de *localité*. Ce besoin particulier de notre système de GAT pose plusieurs difficultés : dans une perspective de génération multilingue nous aurons besoin d'un système d'EEN adaptable rapidement pour une langue donnée. Pour répondre au mieux à des besoins de GAT variés, nous aurons aussi besoin de pouvoir rapidement adapter un jeu d'étiquettes pour répondre à un cadre sémantique de génération particulier (encyclopédique, journalistique, médical) et modéliser ses EN distinctives. Aucun des systèmes disponibles actuellement ne répond à ces particularités, ce qui nous conduira à étudier en profondeur une nouvelle proposition de système d'EEN.

5.3 Détection des co-références

Le phénomène des co-références - illustré par la figure 5.3 - est historiquement étudié par la linguistique et fait l'objet de nombreuses publications dans les communauté de linguistiques et de TAL française et internationale. La co-référence décrit le principe de rattachement de plusieurs expressions à un concept unique (une personne, un objet), étant admis que les expressions à regrouper peuvent répondre à des graphies et des natures syntaxiques différentes. Le problème de la proéminence (*salience*) rend la résolution par approche statistique particulièrement adaptée au regroupement de co-références, lorsque elle est hybridée avec des méthodes à base de règles. L'*algorithme de Hobbs* [8] (Hobbs, 1978) est apprécié pour sa simplicité. Ne recourant qu'à un analyseur syntaxique et à un analyseur de genre, il est prévu pour résoudre les anaphores pronominales. On notera que dans le champ de la GAT, de nombreux auteurs s'attachent à produire (et non détecter) des co-références. On pourra par exemple se reporter à (Dale et Viethen, 2009) sur ce sujet. Nous verrons que dans le cas de l'approche de GAT par apprentissage sur corpus que nous proposons, la co-référence étant pré-existante dans une phrase à ré-employer, c'est surtout au processus de détection

7. Source **Groupe de Travail ESTER 2**, Paris, Janvier 2009 http://www.afcp-parole.org/ester/index.html.

8. Cet algorithme est détaillé dans (Jurafsky et al., 2000) (page 704, 21.6).

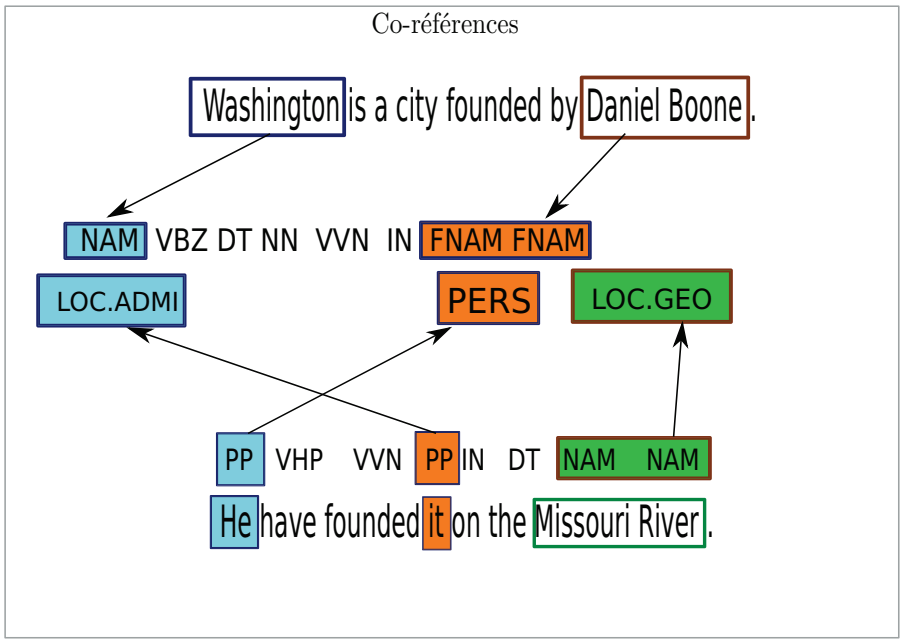

FIGURE 5.3 – *Illustration de la problématique de la détection des co-références.*

que l'on s'attache. Les méthodes de production de co-références décrites peuvent se révéler intéressantes dans notre cadre d'application lorsqu'un processus d'ingénierie inverse peut leur être appliqué. Ce besoin de détecter autant que de produire les co-références (par exemple pour les insérer dans un composant de génération statistique) est d'ailleurs admis par la communauté de recherche de la GAT [9] puisque la campagne GREC (Belz et al., 2009a) est autant une campagne de génération que de détection et de résolution de co-références.

5.3.1 Maturité des technologies

Ce champ de recherche reste relativement ouvert et reçoit régulièrement de nouvelles propositions. Les algorithmes disponibles, conjugués aux améliorations des outils qu'ils impliquent (EEN, parseurs, étiqueteur POS) permettent de déployer des applications exploitables dans un contexte appliqué, sans nécessairement exiger de recherches poussées. Des outils libres de résolution tels que Lingpipe [10] sont disponibles mais rarement en français. Certains phénomènes ou combinaisons de phénomènes demeurent non-triviaux à prendre en compte, comme par exemple les variations de formes de surface des EN impliquées dans plusieurs co-références. Pour succéder à des propositions fonctionnelles mais déjà assez anciennes (Hobbs, 1978), plusieurs campagnes d'évaluation ont vu le jour et tentent de susciter l'émergence de nouvelles propositions. Un travail soutenu a été mené ces derniers années notamment dans le cadre des campagnes

9. Par l'entremise du groupe de recherche SIGGEN de ACL http://www.siggen.org/.
10. Voir http://alias-i.com/lingpipe/.

TUNA (Gatt et al., 2008; Reiter et Gatt, 2009) et plus récemment GREC (Belz et al., 2009b). La dernière campagne d'évaluation GREC a eu lieu en 2010 lors de la conférence INLG (Belz et al., 2009a). Les résultats obtenus dans ces campagnes laissent apparaître une marge de progression encore conséquente. On observe une certaine normalisation : les meilleurs systèmes intègrent progressivement tous des modules d'étiquetage par entités nommées (pour localiser les entités co-référentes) associés à des algorithmes de regroupement de co-références à base de règles plus ou moins sophistiquées (Stoyanov et al., 2010; Charton et al., 2010b; Belz et Kow, 2010).

5.4 Analyse sémantique

L'analyse sémantique (AS) consiste à identifier au sein d'une phrase les relations de sens entre ses constituants en utilisant comme des relations les dépendances comme support d'identification. C'est ce composant - stratégique dans notre architecture - qui peut permettre de représenter les relations de prédicats et variables contenues dans une phrase. En ce sens il est sémantique puisqu'il détecte la relation entre les syntagmes (il saura par exemple indiquer que *Le TGV* et *Lyon* sont reliés par le prédicat *rejoint* soit en logique de premier ordre *rejoint(TGV,Lyon)*. C'est une tâche non triviale du TAL qui a connu un intérêt croissant au cours de la période récente. Elle repose sur une chaîne de traitement composé de l'analyseur sémantique proprement dit, appliqué sur une décomposition des dépendances de la phrase, obtenue après l'application d'un analyseur syntaxique. Cette analyse permet d'identifier les rôles des différents syntagmes et de préparer la tâche de l'analyse sémantique. Lors de l'étape finale, les relations sémantiques identifiées sont transformées en prédicats logiques.

5.4.1 Maturité des technologies

Cette tâche a considérablement évoluée au cours de 5 dernières années, et en particulier grâce à l'influence prépondérante d'une campagne dédiée des *shared task* CoNLL (Surdeanu et al., 2008; Hajič et al., 2009) [11]. À la suite de cette série de campagnes, il est désormais possible de disposer d'analyseur sémantique au niveau de l'état de l'art. Le classifieur le plus performant de la tâche 2008 (Johansson et Nugues, 2008) est disponible librement [12] et permet d'obtenir des analyses sémantiques fiables (F-Score reproductibles de 0,86 obtenu sur le Brown Corpus lors de CoNLL 2008) aisément convertibles en structures logiques du premier ordre. La difficulté subsistante dans notre cadre applicatif est que les analyseurs sémantiques disponibles ne sont pas entraînés pour la langue française. Nous avons pour traiter cette langue appliqué une forme rudimentaire des propositions de (Zouaq et Gagnon, 2010) en utilisant un analyseur de dépendances syntaxique français mis au point pour la campagne ConLL-X (Buchholz et Marsi, 2006).

11. Voir http://ufal.mff.cuni.cz/conll2009-st/ et http://213.27.241.151/dokuwiki/doku.php?id=conll2008:start.
12. Voir http://nlp.cs.lth.se/software/semantic_parsing:_propbank_nombank_frames.

5.5 Conclusion

Nous avons passé en revue les différents niveaux d'étiquetage et d'extraction de connaissances qu'il est possible d'obtenir d'après une phrase. Les méthodes issues du domaine du TAL impliquées dans les divers composants de notre système d'apprentissage ont été passées en revue dans la perspective de mise au point d'un système de création de *Corpus de Phrases Modèles (CPM)* tel que présenté dans le chapitre 4. Nous avons observé qu'un certain nombre de méthodes pouvaient être considérées comme mature en l'état (par exemple l'étiquetage morphosyntaxique), que d'autres bien que fonctionnelles, pouvaient justifier un travail d'amélioration (l'étiquetage par entités nommées ou la détection de co-références). Nous avons observé que certaines techniques, bien que fonctionnelles, pouvaient être difficiles à déployer en contexte multilingue (la décomposition des dépendances d'une phrase et l'analyse sémantique par exemple). Pour élaborer le système de modélisation de phrase prévu par le module d'apprentissage de notre système de GAT, nous explorerons dans les chapitres qui suivent chacun de ces aspects et proposerons des modifications ou des améliorations.

Chapitre 6

Construction de métadonnées d'après un contenu encyclopédique

Sommaire

À de nombreux niveaux de son architecture, un système de GAT à composants statistiques fait appel à des connaissances de nature lexicale. Ces connaissances lexicales permettent par exemple de *sérialiser* un objet textuel pour le rendre plus abstrait. Il existe déjà de nombreuses ressources lexicales disponibles et ré-exploitables pour les noms communs ainsi que les verbes. Les ressources lexicales exhaustives et riches inventoriant les différentes formes d'écritures d'une entité nommée (un nom de société, de produit) sont en revanche beaucoup plus rares. Il en va de même pour certains termes (des noms de technologies, d'objets) ou pour des locutions. Nous proposons dans ce chapitre de bâtir d'après des contenus encyclopédiques une ressource dont le contenu est notamment lexical. Les *métadonnées* que nous allons construire définissent exclusivement les éléments porteurs de sens qu'une phrase est censée contenir. Nous associerons à ces *métadonnées* des informations taxonomiques et statistiques sur les objets qu'elles décrivent. Nous utiliserons ces informations complémentaires par la suite pour entraîner divers composants d'un système d'étiquetage riche de corpus.

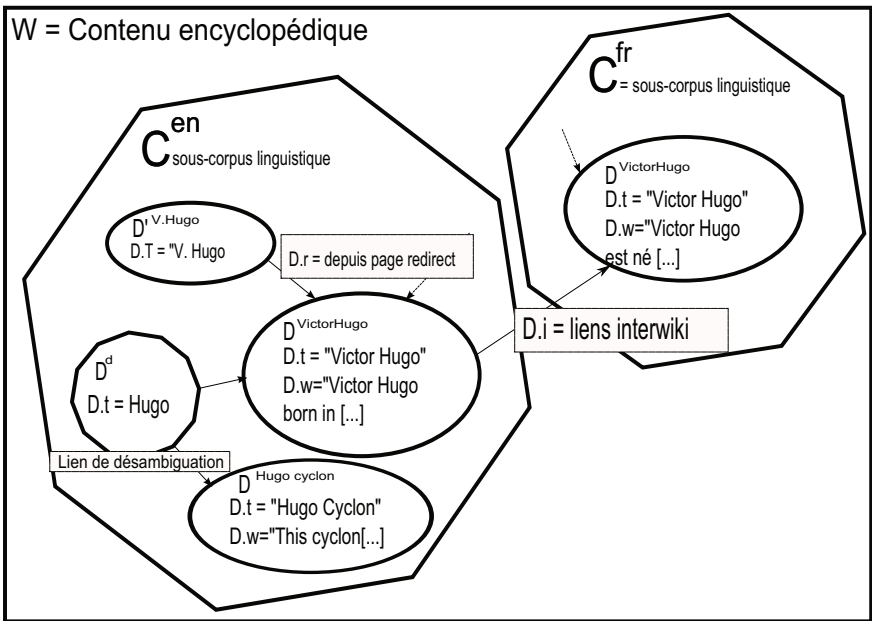

FIGURE 6.1 – *Structure des données dans Wikipédia.*

6.1 Construction de métadonnées d'après Wikipédia

Notre système de construction des *metadonnées* assure une transformation de la structure interne de Wikipédia en une représentation sémantique et statistique. Dans la section 6.1.1, nous définissons la structure interne des corpus Wikipédia. Dans la section 6.1.2 nous décrivons de manière formelle une *metadonnée* : chaque enregistrement de *metadonnées* est composé du nom de l'article encyclopédique utilisé en tant que clé, d'un graphe représentant les formes de surface potentielles d'une *entité nommée* (EN), d'un label représentant une classe sémantique et d'un ensemble de mots et leurs poids $TF.IDF$. Nous expliquons quels éléments d'un article encyclopédique sont utilisés pour construire une *metadonnées*. Puis nous décrivons dans la section 6.1.3, l'algorithme déployé pour procéder à la transformation de Wikipédia en *metadonnées*.

6.1.1 Structure de la ressource encyclopédique Wikipédia

Dans notre application, considérons M qui représente l'encyclopédie Wikipédia et C^l un corpus représentant une version linguistique. Cette version linguistique contient des articles structurés d'après une DTD [1]. Un ensemble d'*espaces de noms* [2] décrit par la DTD permet de reconnaître les articles (contenus dans l'espace encyclopédique) des autres documents (modèles, utilisateurs, images). Chaque article contient en ensemble de mots, en relation avec le concept encyclopédique qu'il décrit.

1. Consulter http://meta.wikimedia.org/wiki/Wikipedia.DTD.
2. Lire http://fr.wikipedia.org/wiki/Aide:Espace_de_noms pour des précisions sur l'espace de nom de Wikipédia.

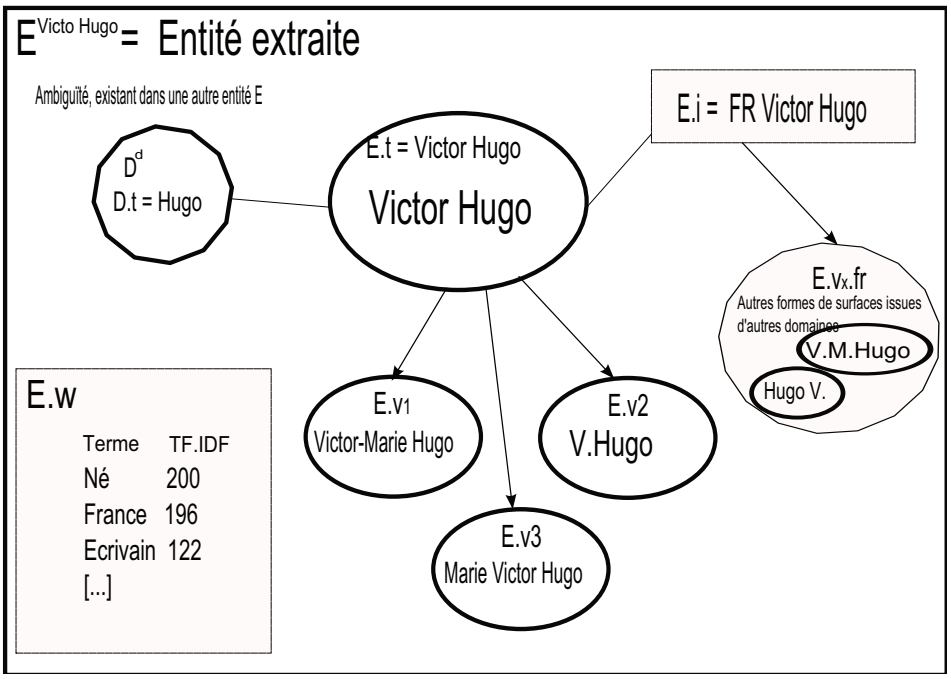

FIGURE 6.2 – *Structure des données d'une metadonnée.*

Considérons D un article du corpus Wikipédia C^l, défini par des propriétés :

- $D.t$ est un titre composé d'une séquence de mots.
- $D.w$ est la liste des mots contenus dans l'article.
- Si un titre d'article est ambigu par homonymie de son titre descriptif, une page spéciale intitulée **Page d'homonymie** dans la version française ou ***Disambiguation page*** dans la version anglaise est créée. Nous intitulerons ces pages $D^d \in C^l$. Elles contiennent plusieurs références à des pages $D \in C^l$ à désambiguïser.
- Des articles uniques dits **Pages de redirections** existent pour répertorier les noms alternatifs susceptibles de correspondre à un article de Wikipédia. Nous utilisons ces pages pour générer les graphes de formes de surface. Nous appelons D^r une *page de redirection* et $D^r.R$ le lien de redirection unique qu'elle contient et qui correspond au nom original $D.t$ de l'article D vers lequel elle redirige.
- Chaque article contenu dans une version linguistique de Wikipédia peut inclure des liens vers les articles similaires contenus dans d'autres versions linguistiques de l'encyclopédie. Ces liens sont intitulés des **Interwiki**. Nous intitulons *relation interwiki* $D.i$, le lien de D vers son équivalent dans un corpus d'une autre langue $C^{l'}$.

6.1.2 Les *metadonnées* produites d'après la ressource encyclopédique

Les *metadonnées* de notre application sont composées d'un jeu d'entités E dérivées de $D \in C^l$. Chaque entité $E \in C^l$ est définie par un jeu de propriétés $(E.t, E.v, E.w, E.i, E.k)$.

- $E.t$ est le titre de l'entité, correspondant à $D.t$, titre unique d'une page Wikipédia.
- $E.v$ est l'ensemble de toutes les formes de surface qui peuvent écrire E. $E.v$ contient donc des formes de surface synonymes. Cet ensemble est construit d'après les *pages de redirection* D^r reliées à la page D utilisée pour construire E. On notera que lorsqu'une *page d'homonymie* D^d existe, contenant des liens vers la page D utilisée pour construire E, le titre $D^d.t$ de D^d est inclus dans $E.v$.
- $E.i$ est l'ensemble de *relations interwiki* contenues dans la propriété $D.i$ correspondante. $E.i$ représente la relation entre $E \in C^l$ et tout $E' \in C^k, k \neq l$. Ceci signifie que $E.i$ contient les références des entités correspondantes dans les autres corpus linguistiques que celui de départ.
- $E.w$ est un ensemble de mots avec leurs poids (exprimés sous la forme d'une valeur $TF.IDF$) associés à l'entité. Cet ensemble de propriétés est construit d'après le texte $D.w$ contenu dans l'article D original de Wikipédia, utilisé pour construire E.

Nous avons besoin d'associer aux entités extraites depuis Wikipédia une classe pour introduire dans les *métadonnées* une information sémantique exploitable ultérieurement pour entraîner des systèmes d'étiquetage.

- Nous ajoutons donc aux *métadonnées* une propriété $E.k$ qui est un label d'étiquetage (exemple *personne, produit, lieu* ou *organisation*) en accord avec le standard ESTER 2 [3]. Une classe spécifique intitulée *unknown* est par ailleurs introduite pour retirer de la liste de détection des EN les articles relatifs à des descriptions encyclopédiques inutiles pour la tâche d'EEN (un théorème mathématique, une locution, un nom commun par exemple).

6.1.3 Transformation d'un article encyclopédique en *metadonnées*

Une description formelle de l'algorithme de construction d'une *metadonnée* d'après un article encyclopédique peut être présentée comme suit :

Considérons que tout article Wikipédia $D \in C^l$ est défini par une ensemble de propriétés $(D.t, D.w, D.r, D.i)$ telles que $D.t$ est un titre représenté par une séquence de mots, $D.w$ un ensemble de mots, $D.r$ une relation unique entre D et n'importe quel élément de C^l, $D.i$ est un ensemble de *relations interwiki* entre des éléments de D et n'importe quel élément de $C^{l'} \in M \backslash C^l$. Pour générer une table T^l d'entités, nous explorons dans un premier temps C^l et conservons tout élément de D qui n'est pas une **Page de redirection** (D^r) ou une **Page d'homonymie** (D^d). On considère que E et D sont en relation si et seulement si $E.t = D.t$. Ceci est formalisé par $E \rightarrow D$. Puis, pour tout E de la table T^l, nous cherchons tous les $D^d \in C^l$ en relation avec E et incluons le nom alternatif $D^d.t$ dans $E.v$. Nous cherchons aussi tous les $D^d \in C^l$ en relation avec E et définissons $E.v = D^d.t$.

Ces étapes sont répétées pour les corpus de toutes les versions linguistiques de

3. Voir la convention d'annotation sur http://www.afcp-parole.org/ester/docs/Conventions_EN_ESTER2_v01.pdf.

Wikipédia retenues. Nous obtenons donc pour chaque corpus linguistique de Wikipédia C^l une entité E^l. Puis, pour procéder à l'agrégation des formes de surface localisées dans ces différentes versions, nous collectons les *relations interwiki* disponibles dans la section $D.i$ de l'article encyclopédique et les attribuons à la propriété $E.i$. Nous agrégeons toutes les formes de surface $E^{l.v}$ dans l'entité $E.v$.

6.1.4 Exemple de transformation d'un article en *metadonnées*

La structure d'une *metadonnée* $E.$*Victor Hugo* construite d'après l'article $D.$*Victor Hugo* est présentée dans la figure 6.2. Les informations originales D contenues dans Wikipédia ayant servi à construire E sont illustrées dans la figure 6.1. Pour construire E, considérons que le titre $D.t$ =*Victor Hugo* de la page Wikipédia est utilisé en tant que descripteur unique $E.t$ de la *metadonnée Victor Hugo*. Les formes de surface sont ensuite collectées pour construire le graphe $E.v$. D'abord, nous ajoutons le titre original $D.t$ à $E.v$. Ce titre encyclopédique original est toujours la forme minimale (et parfois unique) d'écriture possible pour $E.v$. Les **Pages de redirection** de Wikipédia D^r contenant un lien $D^r.R$ vers D =*Victor Hugo* sont recherchées ; chaque titre $D^r.t$ d'une **Page de redirection** est un synonyme de *Victor Hugo* et est inclus dans $E.v$. Nous cherchons les **Page d'homonymie** D^d qui contiennent des liens vers l'entité D. Ces liens sont contenus à l'intérieur du corps de texte de D^d en accord avec les spécifications du formalisme de Wikipédia. Dans l'exemple de *Victor Hugo*, la **Page d'homonymie** D^d est «Hugo» et est reliée à 45 autres pages encyclopédiques qui décrivent des personnes (la page *Victor Hugo* elle-même), des lieux, des événements (ouragans) ou des organisations (un prix littéraire). Nous ajoutons le titre $D^d.t = Hugo$ à $E.v$.

L'agrégation finale de toutes les formes de surface est réalisée par l'exploration des *relations interwiki*. Les formes collectées sont décrites dans la figure 6.2 par le lien $E.i$. On peut observer une introduction de bruit lors de la collecte des formes de surface. En effet le graphe de formes de surface de *Victor Hugo* contient 14 formes valides et une forme erronée, *Adèle Foucher*. Ce phénomène est lié au caractère manuel de la saisie des *pages de redirection* par les contributeurs de Wikipédia. Nous discutons de son influence dans nos expériences. Tous les termes de l'article *Victor Hugo* de la version linguistique française de Wikipédia font l'objet d'un calcul de poids $TF.IDF$ (d'après l'intégralité de C^l) et sont intégrés dans $E.w$. On observe que les termes de poids les plus forts sont hautement contextuels (*Écrivain*, *France*, etc.) [4].

6.2 Classification

Nous avons développé un ensemble d'algorithmes de classification permettant d'extraire et de structurer les contenus encyclopédiques, mais aussi de transformer la classification interne très complexe et ouverte de Wikipédia. Considérons, pour la mise au

4. Voir http://www.nlgbase.org/perl/display.pl?query=VictorHugo&search=FR pour un descriptif complet.

point de notre système, une entité encyclopédique et ses éléments constitutifs :

- Elle contient un texte qui peut être analysé par un classifieur numérique (voir figure 6.3).
- Des *Catégories* lui sont attribuées, faisant partie d'un graphe qui peut être exploré et partiellement décrit par un arbre taxonomique (voir figure 6.4).
- Elle contient des boîtes d'information nommées *Infobox* qui décrivent des propriétés standardisées de l'objet décrit par l'entité encyclopédique. Par exemple, pour un écrivain, sa date de naissance, son lieu de naissance, etc. (voir figure 6.3)

FIGURE 6.3 – *La fiche de Victor Hugo dans Wikipédia contient un texte descriptif et une boîte d'information (sur la droite). Dans le cas de cette fiche, la boîte d'information est celle des écrivains.*

Catégories : Bon article | Victor Hugo | Besançon | Personnalité du XIXe siècle | Écrivain français du XIXe siècle | Écrivain romantique | Poète français du XIXe siècle | Poète romantique français | Dramaturge français du XIXe siècle | Opposant au Second Empire | Philhellène | Personne inhumée au Panthéon de Paris | Membre de l'Académie française | Naissance en 1802 | Naissance à Besançon | Décès en 1885 | Décès à Paris | Naissance dans le Doubs | [+]

FIGURE 6.4 – *Les catégories associées à la fiche Victor Hugo de Wikipédia.*

Notre idée est de tirer parti de l'existence de ces trois éléments susceptibles de participer d'une classification. Dans un tel cadre applicatif, nous nous trouvons face à des ensembles ayant des intersections plus ou moins larges, suivant des règles à finalités divergentes. Notre système fonctionne en deux phases que nous intitulerons W_k1 et W_k2 :

Phase W_k1 : première classification

La **première phase** permet de créer une classification générale, du niveau de l'état de l'art pour ce qui est de la classe racine (par exemple la classe descriptive des *Organisations*, ORG), et moins exhaustive et précise pour ce qui est du second niveau d'étiquetage de l'arbre taxonomique (ORG.DIV ou LOC.GEO). La phase W_k1 se déroule ainsi :

- Une étape de classification numérique est appliquée pour inventorier les contenus affectés aux classes racines (par exemple ORG, PERS, LOC ou PROD).
- Une étape de classification est menée à partir de règles établies d'après des catégories sélectionnées au sein de la taxonomie de Wikipédia, pour le second niveau de classe (par exemple ORG.DIV, LOC.ADMI ou PERS.HUM).
- Une étape de classification est conduite à partir de règles établies d'après des catégories pour les micro-classes (tel que les titres nobiliaires ou les fonctions).

A l'issue de cette première phase, nous obtenons une première classification qui décrit le second niveau taxonomique très partiellement. Par exemple 85% à 89% des lieux sont correctement étiquetés LOC, mais moins de la moitié des éléments de sous classes LOC.GEO ou LOC.ADMIN est détectée. Il faut donc affiner le processus.

Phase W_k2 : perfectionnement de la classification

Nous utilisons les informations déjà disponibles pour procéder à un nouvel apprentissage suivi d'une **nouvelle phase** de classification, la phase W_k2 :

- L'apprentissage consiste à inventorier la classe attribuée lors de la phase précédente pour chaque document Wikipédia contenant une *Infobox*.
- Cette information étant disponible, la probabilité qu'une classe d'étiquetage soit associée à une *Infobox* est évaluée. Puis des règles d'association entre *Infobox* et classes sont élaborées (par exemple *Infobox Communes de France* sera associée à LOC.ADMI).
- Le corpus Wikipédia est reclassé en utilisant en tant que règles d'attribution de classe la présence d'une *Infobox*.

La seconde phase de détection W_k2 s'applique à la totalité des pages contenant une *Infobox* [5]. A l'issue de cette phase, les classes associées aux *Infobox* sont fiables à 99%. Après cette phase, nous obtenons un gain de qualité finale de la classification - par rapport à W_k1 - de l'ordre de 8% à 15% (voir tableau 6.4).

6.2.1 Classification numérique

Le système numérique utilisé pour W_k1 est une fusion ternaire des résultats produits par un classifieur SVM, un classifieur bayésien naïf et AdaBoost (Charton et al., 2008).

5. Mesure réalisée sur le dump XML Wikipédia référencé frwiki-20081201-pages-articles.xml disponible sur http://download.wikipedia.org.

Les classes de détection sont construites après application de pré-traitements au corpus d'apprentissage : linéarisation des classes par comparaison des distributions de Zipf, utilisation de 3-grammes et application d'un anti-dictionnaire.

La fusion par vote majoritaire que nous avons déployée est triviale. Elle consiste à confronter les propositions des trois meilleurs classifieurs (SVMLib, Naïve Bayes, Icsiboost) pour chaque document. Si une majorité l'emporte (2/3 ou 3/3), la classe majoritaire est choisie, dans le cas contraire, la stratégie de fusion se replie sur le système le plus performant (le classifieur SVM avec noyau linéaire).

Ce système n'est efficace que sur des classes de poids équivalents et n'est donc déployé que sur les classes PERS, PROD, ORG, LOC et UNK. Les classes FONC et DATE (représentant moins de 1% du corpus) ne sont pas modélisables avec ce système de classification, tout comme les sous-classes.

6.2.2 Classification d'après des *Infobox*

Pour exécuter la phase W_k2 nous utilisons les *Infobox* de Wikipédia afin d'introduire un second niveau de classement des entités encyclopédiques.

Soient un ensemble D de documents issus de Wikipédia, un ensemble C de catégories de Wikipédia et un ensemble I d'*Infobox* de Wikipédia. Dénommons *étiquette de second niveau (ESN)* la sous-classe d'une classe racine.

Considérons un ensemble de documents $E \in D$ appartenant à une catégorie $c \in C$, un ensemble de documents $F \in D$ munis d'une *Infobox* $i \in I$, et un ensemble de documents $G \in D$ étiquetés par un label l qui est une ESN.

Soit $U = E \cap F \cap G$.

Tous les éléments de U ont en commun la catégorie c, l'*Infobox* i et le label d'ESN l. On peut donc en déduire une association directe entre l'*Infobox* i et le label d'ESN l pour les documents de U.

Ceci nous permet d'élaborer automatiquement, en partant d'un petit groupe de catégories représentatives de chaque ESN, la table des associations entre les *Infobox* et les étiquettes ESN. Nous utilisons cette table d'associations pour reclasser toutes les entités de Wikipédia en détectant la présence éventuelle d'une *Infobox* dans le document qui les décrit et en lui attribuant le label d'ESN qui lui est associé.

6.2.3 Classification d'après des catégories

Un résidu de documents du corpus Wikipédia ne peut être classé par les deux méthodes précédentes : soit leur contenu est trop faiblement informatif pour appliquer la classification numérique, soit ils ne contiennent pas d'*Infobox*. Pour ces documents, nous prévoyons dans la phase W_k2 une dernière étape qui consiste à associer une étiquette à une catégorie de Wikipédia. Cette méthode peut être très performante

si une classe est fortement représentative, mais très coûteuse lorsque les catégories sont mal définies ou très granulaires. On notera, par exemple, que plus de 144.000 entités encyclopédiques sont associées à la catégorie *Naissance en* : ce type de catégorie utilisé comme règle de détection augmente considérablement la précision de la classe PERS.HUM. En revanche, la catégorie *Locution ou expression latine* qui devrait catégoriser des entités de type UNK est de faible utilité avec les 124 éléments encyclopédiques qu'elle contient [6]. Les règles catégorielles, du fait de leur variabilité, ne peuvent donc être employées que pour affiner ou renforcer l'étiquetage des entités ou pour traiter des micro-classes non modélisables par méthodes numériques.

6.2.4 Expériences et résultats

Pour évaluer l'intérêt de notre dictionnaire d'entités nous avons procédé à un ensemble d'expériences et de mesures. Les premières mesures de performance sur le système de classification numérique ont été réalisées lors de la campagne d'évaluation de classification DEFT 2008 (Charton et al., 2008). La campagne DEFT 2008 (Grouin et al., 2008) (DÉfi de Fouilles de Texte [7]) avait pour sujet la classification en genre et en thème de textes. Les corpus proposés pour les deux tâches ont été constitués à partir de deux sources distinctes : Le Monde et la version française de Wikipédia. Le premier corpus, noté Corpus1, est étiqueté en thème selon quatre classes : économie (ECO), arts (ART), télévision (TEL) et sport (SPO). Chaque document est également annoté selon qu'il provient du journal Le Monde (LM) ou de Wikipédia (W). Le corpus de la tâche 2, noté Corpus2, est composé de documents annotés en thèmes selon cinq classes : société (SOC), actualité ou information française (FRA), ou internationale (INT), sciences (SCI) et littérature (classe LIV).

Nous évaluons par ailleurs la qualité de l'étiquetage selon le standard établi par la campagne Ester 2 pour ce qui est de la taxonomie des entités nommées et de la vérification de couverture du dictionnaire. La qualité de la classification des entités de Wikipédia obtenues avec les algorithmes W_k1 et W_k2 est mesurée par la précision et le rappel d'après un échantillon de référence. Seules les entités racines et de second niveau des familles PERS, ORG, LOC, PROD et FONC ont été retenues pour ces expériences. Les étiquettes de type AMOUNT, TIME, ne sont pas concernées par le dictionnaire extrait de Wikipédia et ne sont donc pas mesurées ici. Les étiquettes de type DATE qui sont présentes en tant que descriptifs de dates historiques dans Wikipédia, sont mesurées à titre indicatif mais ne sont pas exploitées dans le cadre applicatif de la détection.

Mesure du classifieur numérique W_k1 lors de la campagne DEFT 2008

L'une des difficultés de la tâche était l'existence de classes sous-représentées qui perturbent l'apprentissage automatique. On observe dans Corpus1 l'existence de classes dont la répartition est très disparates. Ce déséquilibre est conjugué à des imbrications

6. Voir les analyses de catégories sur www.nlgbase.org/fr/stat/stat_cat.html.
7. Voir http://deft08.limsi.fr.

Fscores APP	tâche1-catégorie	tâche1-genre	tâche2-catégorie
SVM_baseline	0,8391	0,93	0,78
SVM_extended	**0,9150**	0,9594	**0,8445**
N_Bayes_extended	0,8629	0,9353	0,8271
BoosTexter	0,8958	0,9869	0,8316
Isciboost	0,9051	**0,9858**	0,8409
Cosinus	0,8508	0,9222	0,8244

TABLE 6.1 – *F-Scores de tous les classifieurs testés sur les données de DEFT 2008.*

Exécution 1	Précision	Rappel	F-Score
tâche1-genre	0,9795	0,9800	0,9798
tâche1-catégorie	0,9082	0,8448	0,8754
tâche2-catégorie	0,8814	0,8759	0,8786

TABLE 6.2 – *Résultats du vote par fusion ternaire majoritaire des classifieurs SVM, Isciboost et bayésien naïf.*

inter-classes importantes [8]. L'imbrication est particulièrement marquée dans le cas des classes TEL et ART. La confusion entre ces deux classes est conjuguée à une faible représentation de TEL par rapport à ART (respectivement 8.88% et 37.88% des documents). Il apparaît que contrairement à Corpus1, Corpus2 repose sur une répartition des classes plus homogène. On note, à nouveau, une légère sous-représentation d'une classe (FRA) associée à une sur-représentation d'une autre (SCI). Les trois classes FRA, SCI et SOC ont par ailleurs des intersections fortes, pouvant conduire à une faiblesse de rappel sur les classes FRA et SOC et un manque de précision très marqué sur SOC.A titre de comparaisons, plusieurs méthodes de classification ont été testées. Nous présentons dans le tableau 6.1 l'ensemble des meilleurs F-Scores obtenus par chaque classifieur sur les trois tâches de DEFT 2008, lors de la validation croisée. Les résultats par fusion par vote ternaire de classifieurs SVM, bayésien naïf et Boostexter sont les meilleurs obtenus sur les trois exécutions (indiqués dans le tableau 6.2). On observe un maintien des performances sur tâche1-genre et tâche2-catégorie et une baisse de performance assez marquée sur la tâche1-catégorie, explicable par la différence de répartition des classes entre les corpus d'apprentissage et d'évaluation. Les résultats finaux obtenus lors de la campagne sont indiqués dans le tableau 6.3.

Mesure de la classification complète W_k1 et W_k2 selon la taxonomie Ester 2

Les résultats de l'attribution de classes d'étiquetages aux entités nommées représentées par les entités encyclopédiques de Wikipédia sont présentés dans le tableau 6.4. Les données de référence utilisées pour mesurer la précision et le rappel sont fournies par un corpus de 5.500 entités extraites aléatoirement dans Wikipédia ; 4.800 entités de référence sont étiquetées de manière semi-automatique et corrigées à la main ; 700 entités de référence sont entièrement étiquetées à la main.

8. Les distances entre classes ont été mesurées par similarité cosinus.

Équipe	Rang	tâche1-genre	tâche1-catégorie	tâche2-catégorie
LIA-Charton	1	0,980	0,875	0,872
LIA-Torres	1	0,981	0,883	0,879
LIP-6	1	0,976	0,894	0,876

TABLE 6.3 – *Résultats finaux de la campagne DEFT 2008.*

Classes	W_k1 : Classification numérique			W_k2 : Classification complète		
	Précision	Rappel	F-Score	Précision	Rappel	F-Score
PERS	0,92	0,91	**0,92**	0,95	0,97	**0,96**
ORG	0,74	0,83	**0,79**	0,82	0,89	**0,87**
LOC	0,85	0,89	**0,87**	0,93	0,96	**0,95**
PROD	0,80	0,94	**0,86**	0,90	0,95	**0,93**
UNK	0,95	0,78	**0,85**	0,96	0,92	**0,94**
DATE	-	-	-	1	1	**1**
FONC	-	-	-	1	1	**1**
Total			**0,86**			**0,95**

TABLE 6.4 – *Précision, Rappel, F-Score (obtenus sur le corpus de test.*

Le tableau 6.4 est divisé en deux sections : la première présente le F-Score [9] obtenu pour chaque classe par la classification numérique de l'algorithme W_k1 ; la seconde met en évidence l'amélioration obtenue après utilisation des modes de détection de classes complémentaires offerts par les classifieurs de l'algorithme W_k2. Les scores détaillés des sous-classes introduites dans l'algorithme W_k2 ne sont pas présentés ici [10]. En effet, seules les classes racines sont classifiées à la fois par W_k1 et W_k2 ce qui permet une comparaison des résultats obtenus (les sous-classes n'étant classifiées que par W_k2).

À l'exception de la classe ORG qui pose des problèmes de modélisation spécifiques [11], la précision et le rappel obtenus sur des entités classées semblent suffisamment élevés (de l'ordre de 95% de précision) pour envisager une utilisation dans une tâche d'EEN.

6.3 Conclusion

Nous avons présenté un système capable de produire, d'après un corpus encyclopédique tel que Wikipédia, un dictionnaire multilingue de concepts que nous avons intitulé *métadonnées*. Ces *métadonnées* incluent des noms propres, des noms communs, des entités nommées, et des locutions rigoureusement classées et associées chacune à plusieurs formes d'écriture possible. Ce corpus d'entités et de concepts labelisés permet d'extraire facilement des sous-lexiques spécialisés de noms propres, de noms de

9. Mesure harmonique combinant la précision et le rappel.
10. Voir le détail des sous classes sur www.nlgbase.org/fr/stat/stat_clust.html.
11. La classe ORG prévue par la campagne ESTER 2 fait cohabiter des partis politiques, des organisations commerciales, des entités géopolitiques et des émissions de divertissements. Ces entités sont suffisamment différentes pour rendre délicate la mise au point de leur classe de détection.

personnes, de lieux, ou encore des locutions nominales, qui seront utilisés dans les composants de lexicalisation du système de GAT que nous proposons d'élaborer. Ces *métadonnées* pourront être utiles dans une tâche d'EEN associées par exemple à une mesure de similarité pour désambiguïser des entités en contexte ou encore pour améliorer l'apprentissage d'un étiqueteurs.

Chapitre 7

Étiquetage par entités nommées

Sommaire

Lors de la modélisation de phrases, nous faisons intervenir un processus d'étiquetage qui rend les entités susceptibles d'être réemployées dans un processus de GAT, les plus abstraites possible. Ceci signifie notamment que dans une phrase candidate au réemploi, les éléments substituables doivent être remplacés par une forme symbolique. Le nom d'une personne, d'un lieu, d'une organisation, une date ou une quantité font partie de ces éléments aisément substituables. Ils doivent donc être détectés et marqués en vue de leur remplacement. L'EEN est la tâche d'extraction d'information qui correspond à ce besoin. Les entités détectées correspondent aux principaux concepts de base qui peuvent être trouvés dans un document. Depuis la campagne d'évaluation MUC, suivie par CoNLL (Tjong et Meulder, 2003) et ACE, de nombreuses méthodes ont été proposées et évaluées pour entraîner des systèmes d'EEN fonctionnant par apprentissage automatique sur des corpus annotés. Ces corpus proposent des taxonomies d'étiquetage qui varient en fonction de l'objectif envisagé par la campagne (Charton et Torres-Moreno, 2009). L'entraînement des systèmes peut par ailleurs être conditionné par la nature du texte à étiqueter. Certains campagnes comme CoNLL s'appliquent à

soumettre à l'étiquetage des corpus issus de publications dans lesquelles de nombreux points de repères typographiques peuvent être utilisés, telles que les lettres majuscules et minuscules. D'autres - telle que les campagnes ESTER - proposent d'appliquer un étiqueteur sur des sorties de système de dialogue très bruités, dans lesquelles ces points de repère disparaissent.

L'utilisation de lexique d'entités, de plus en plus fréquemment issus de corpus tels que Wikipédia (Bunescu et Pasca, 2006; Nothman et al., 2009; Kazama et Torisawa, 2007) est une solution applicable pour rechercher et étiqueter ces entités mais insuffisante pour plusieurs raisons. En premier lieu de nombreuses entités à détecter sont absentes de ces corpus de ressource (fussent ils aussi vastes que Wikipédia) selon le principe des *mots hors vocabulaire* (OOV) ; en second lieu, de nombreuses *entités nommées* (EN) sont hautement ambiguës (par exemple Paris qui est un nom de personne, de plusieurs villes, de paquebot) et ne permettent pas une détection efficace sans utiliser une analyse du contexte.

Dans ce chapitre, nous présentons un système d'EEN, entraîné par méthode automatique, initialement conçu pour fonctionner de manière robuste avec des corpus bruités issus de transcription automatiques de la parole, mais applicable à notre problématique d'étiquetage de *phrases modèles*.

La première implémentation décrite dans ce chapitre, a été mis au point dans la perspective de la campagne d'évaluation ESTER 2 (Galliano et al., 2009) où elle a obtenu les meilleurs résultats. Nous la qualifierons de **baseline** au sens où elle a servi à valider et évaluer les principes d'un système d'étiquetage appris entièrement automatiquement, et adaptable facilement dans une autre langue. Son principe est de collecter par des méthodes non supervisées toutes les entités contenues dans un lexique issu d'un système de reconnaissance automatique de la parole (RAP). Cette connaissance est exploitée de deux manières : en extrayant automatiquement les EN contenues dans un grand corpus de données et en exploitant directement les *métadonnées* issues de Wikipédia. Son intérêt dans notre cadre applicatif est qu'il utilise les *métadonnées* collectées dans le chapitre précédent pour augmenter la capacité inférentielle du système statistique qu'il utilise. Le module d'EEN mis au point combine une approche générative à base de Modèles de Markov Cachés (HMM) et un modèle discriminant à base de Champs Conditionnels Aléatoires (*Conditional Random Field*) (CRF).

La seconde implémentation décrite est qualifiée de **multilingue**. Elle reprend les grands principes du système *baseline* mais substitue au processus de conception de corpus d'apprentissage par HMM une méthode utilisant les *métadonnées* extraites depuis Wikipédia, en association avec les liens internes contenus dans cette encyclopédie. Cette méthode permet d'extrapoler automatiquement des corpus d'apprentissage étiquetés, utilisés pour entraîner le modèle discriminant à base de CRF. Elle simplifie ainsi le processus de production d'un étiqueteur d'EN pour une langue donnée sans perte de performance.

7.1 Système *baseline*

De nombreuses méthodes ont été proposées pour extraire des EN depuis un texte : des plus simples basées sur des règles, aux plus complexes, reposant sur un apprentissage automatique recourant à des corpus étiquetés. Au fil des années, deux approches principales ont fini par se dégager : les méthodes génératives, à base de HMM (Bikel et al., 1999) et celles de nature discriminantes telles que le *Maximum Entropy* (Borthwick et al., 1998) ou les CRF (McCallum et Li, 2003). On se reportera à la section 5.2 pour une étude sur cette question.

7.1.1 Méthodes hybrides génératives et discriminantes pour l'extraction d'EN

Pour les méthodes d'étiquetage à base d'apprentissage sur corpus, le système d'EEN est vu comme un processus d'étiquetage, par lequel un label est attribué à chaque mot d'une phrase. En étiquetant avec le même label plusieurs mots consécutifs, il est possible de déterminer qu'une entité est décrite par plusieurs mots, y compris si le processus d'étiquetage considère chacun des mots isolément. Deux types de caractéristiques peuvent être utilisées pour prédire l'étiquette d'EN d'un mot w_i :

- on pourra observer les caractéristiques du contexte tels que *le mot précédent :* w_{i-1} ou *le mot suivant :* w_{i+1}, si le mot w_i *débute avec une lettre capitale*, ou si le mot w_{i-1} *est un signe de ponctuation*, ...
- on pourra aussi utiliser des informations *a priori* sur w_i, telles que w_i *est un nom de ville*, ou w_i *est un prénom*, ...

On notera que dans le cas de contextes bruités, tels que les transcriptions issues de système de RAP, les caractéristiques du contexte peuvent devenir rapidement inutilisables que ce soit pour des raisons d'erreurs de transcription ou par défaut de respect des usages typographiques. Un système de RAP par exemple, retourne souvent un flux de texte sans ponctuation et sans lettre capitale. Pour notre cadre applicatif de la GAT, ce problème du bruitage des transcriptions est intéressant car il éprouve le degré de robustesse d'une système d'EEN.

Pour étiqueter des sorties de systèmes de RAP, plusieurs méthodes dédiées ont été proposées. Par exemple en associant aux mots décodés le score de confiance de chaque mot retourné lors du processus de RAP (Sudoh et al., 2006). Une autre méthode - proche de celle que nous allons développer pour les deux systèmes présentés dans ce chapitre - consiste à accroître le poids des connaissances *a priori* intégrées au système d'EEN. Ces connaissances peuvent être intégrées lors du processus d'apprentissage, pour améliorer *à postériori* les capacités d'inférences des modèles HMM ou CRF appris. Il a été montré à de nombreuses reprises que cette connaissance *à priori* pouvait être obtenue depuis des dictionnaires, extraits depuis des ressources textuelles structurées telles que Wikipédia (Kazama et Torisawa, 2007), des corpus textuels bruts (Hori et Nakamura, 2006).

Nous proposons dans le système *baseline* une approche mixte d'EEN reposant en premier lieu sur un processus génératif à base dans un premier temps de HMM chargés

de prédire des étiquettes sémantiques et syntaxiques pour chaque mot d'une phrase. Dans un second temps nous utilisons en tant que processus discriminant des classifieurs à base de CRF, utilisés pour identifier les étiquettes d'entités nommées en utilisant le contexte des mots à étiqueter. On attend de cette méthode qu'elle s'assure que l'ambiguïté des mots soit levée autant que possible par les étiquettes fournies par l'étiqueteur HMM, avant d'entraîner le classifieur CRF.

L'étiqueteur HMM utilisé pour ce système est un étiqueteur morphosyntaxique complété par une fonction d'affectation d'étiquettes sémantiques aux noms propres. Il introduit ainsi un niveau intermédiaire d'identification entre le mot et l'entité nommée étiquetée. Les étiquettes sémantiques sont simplifiées et ne concernent que les *personnes, organisations, lieux et produits*. Pour prédire la meilleure séquence des étiquettes $t_{1,n}$ sur une séquence de n mots $w_{1,n}$ (retournés par $\tau(w_{1,n})$), nous utilisons l'équation 7.1.

$$\tau(w_{1,n}) = arg \max_{t_{1,n}} P(t_{1,n}, w_{1,n}) \tag{7.1}$$

En définissant des termes tels que $t_{1,0}$ et leur probabilité d'apparition, nous obtenons l'équation générale du *Part Of Speech* (POS) indiquée en 7.2.

$$\tau(w_{1,n}) = arg \max_{t_{1,n}} \prod_{i=1}^{n} P(t_i|t_{i-2,i-1}) P(w_i|t_i) \tag{7.2}$$

Dans 7.2, la valeur de $P(w_i|t_i)$ est obtenue directement par le critère de maximum de vraisemblance en calculant $C(w_i, t_i)/C(t_i)$ où $C(w_i, t_i)$ est égal au nombre de fois où w_i est associé à t_i dans le corpus d'entraînement et où $C(t_i)$ est le nombre de fois où sont rencontrés les mots étiquetés avec t_i dans le corpus d'entraînement. En collectant le décompte des diverses ressources textuelles telles que présentées dans les sections 7.1.2 et 7.1.3, nous pouvons facilement modéliser l'ambiguïté sémantique d'un nom propre donné. On notera que le modèle de positionnement BIO est adopté (*Begin*, *Inside*, *Outside*). A titre d'exemple, nous obtenons dans les comptages menés lors de nos expériences les valeurs de quantités suivantes pour le nom propre *Marseille* : LOC=*32973*, ORG=*15731*, PERS=*1140*, PROD=*317*.

La valeur correspondant à $P(t_i|t_{i-2,i-1})$ est obtenue d'après un modèle de langage 3-grammes appris sur les étiquettes de POS et sémantiques d'un corpus annoté automatiquement selon la méthode expliquée dans la section 7.1.2. Le classifieur CRF est finalement entraîné avec les sorties de l'étiqueteur HMM appliqué à un corpus d'entraînement. Un exemple de corpus d'entraînement est donné dans la table 7.1. Le classifieur utilisé est *CRF++* [1].

1. Logiciel CRF++ disponible sur http://crfpp.sourceforge.net/.

Mot	*POS+étiquette sémantique*	*EN+position*
bonjour	NMS	O
investiture	NFS	O
aujourd'hui	ADV	B-TIME
à	PREPADE	O
bamako	XLOC	B-LOC
mali	XLOC	B-LOC
du	PREPDU	O
président	NMS	B-FONC
amadou	XPERS	B-PERS
toumani	XPERS	I-PERS
touré	XPERS	I-PERS
réélu	VPPMS	O
en	PREP	B-TIME
avril	NMS	I-TIME
dernier	AMS	I-TIME

TABLE 7.1 – *Exemple du corpus d'entraînement pour le classifieur CRF tel que produit par l'étiqueteur HMM avec ses étiquettes de Part Of Speech et ses étiquettes sémantiques, associées à la position de l'étiquette d'entité nommée à prédire.*

7.1.2 Mise à jour du modèle d'étiquetage avec un corpus non étiqueté de grande taille

À ce stade nous utilisons une collection importante (1,3 go) de corpus textuels pour extraire des paires de types *(Nom propre, label sémantique)* afin d'estimer $P(w_i|t_i)$. Pour initialiser le processus, nous avons besoin en premier lieu d'un lexique de noms propres fréquents, associés à leur label sémantique et un corpus d'entraînement manuellement étiqueté avec les étiquettes d'EN pour entraîner le modèle CRF. Le processus suit les étapes ci-dessous :

1. Un étiqueteur morphosyntaxique (incluant un lexique de noms propres) est appliqué au corpus d'entraînement.
2. Tous les noms propres non présents dans le lexique de noms propres sont étiquetés avec le label *unknown*.
3. Le modèle CRF est entraîné d'après ce corpus désormais étiqueté avec les marqueurs morphosyntaxiques et les étiquettes sémantiques.
4. Ce premier modèle est appliqué au corpus de grande taille.
5. Dans ce premier corpus étiqueté automatiquement, chaque nom propre w_i correspondant à une EN de type π est étiqueté avec le label correspondant à π.
6. Nous collectons le nombre de paires (w_i, π), et ne conservons que celle dont la fréquence est supérieure à α [2] et nous utilisons ce nombre pour mettre à jour le modèle $P(w_i|t_i)$ de l'étiqueteur HMM.
7. Le processus reprend à 1 après un nombre d'itérations fixées à l'avance.

2. Nous avons défini $\alpha = 5$ lors de nos expériences.

7.1.3 Introduction de métadonnées

EN	Corpus de test	Équivalence dans les *métadonnées*	Couverture(%)
PERS	1096	483	44%
ORG	1204	764	63%
LOC	1218	1017	83%
PROD	59	23	39%
FONC	424	151	36%

TABLE 7.2 – *Couverture des entités contenues dans les métadonnées par rapport aux EN apparaissant dans le corpus d'évaluation d'ESTER 2.*

Nous complétons cette méthode en introduisant les *métadonnées* collectées selon la méthode décrite dans le chapitre 6. Nous avons vu que chaque *métadonnée* incluait un graphe de toutes les formes de surface collectées dans Wikipédia. Par exemple, le graphe pour la *métadonnée Paris* contient 39 formes de surface (eg. *Ville Lumière*, *Ville de Paris*, *Paname Capitale de la France*, *Département de Paris*). Chaque graphe correspond à une entité encyclopédique et est associé à une étiquette correspondant à l'arbre taxonomique de la campagne ESTER 2. Dans l'expérience décrite ici, le nombre d'entités encyclopédiques était de 600 389. [3]. Les données sont extraites de l'édition française du corpus Wikipédia [4].

Nous estimons le nombre de (w_i, π) pour chaque nom propre w_i correspondant à une forme de surface d'une entité encyclopédique et mettons à jour le modèle $P(w_i|t_i)$ de l'étiqueteur HMM. Pour mesurer l'influence de ces éléments, nous avons évalué la couverture des formes de surfaces contenues dans les *métadonnées* par rapport au corpus d'évaluation de la campagne ESTER 2. Ces résultats sont indiqués dans le tableau 7.2.

7.1.4 Expériences et résultats

Ce système à été testé lors de la campagne ESTER 2 (Galliano et al., 2009). La campagne d'évaluation ESTER 2 a été organisée en 2009 conjointement par l'association Française de la Communication Parlée (AFCP) et la DGA, avec la collaboration de l'agence de distribution de ressources ELDA. Cette campagne était divisée en trois tâches, segmentation, transcription et étiquetage d'entités nommées. La tâche d'EEN était elle même divisée en 4 sous tâches, la première consistant à étiqueter un système corrigé manuellement (absence d'erreur de transcription) dit *ref trancript*. Les trois autres tâches recouraient à des systèmes contenant des erreurs de transcriptions d'importance croissante afin de mesurer la robustesse des étiqueteurs d'EN.

3. Le nombre d'entités diffère du nombre total de pages présentes dans le corpus original de Wikipédia en raison de la présence dans ce corpus de pages spéciales telles que les redirections, les pages de désambiguïsations, qui sont incluses dans une *métadonnée* unique.

4. Référence du Dump frwiki-20080323-pages-articles.xml repris depuis http://download.wikipedia.org.

Système/WER	*WER=0*	*WER=12,1*	*WER=17,8*	*WER=26,1*
LIA	23,9	**43,4**	**51,6**	**56,8**
sys2	30,9	45,3	55,5	61,2
sys3	37,1	54,0	60,4	65,2
sys4	33,7	50,7	80,8	82,9
sys5	35,0	55,3	86,5	88,6
sys6	9,9	44,9	60,7	66,2
sys7	**9,8**	44,6	X	X

TABLE 7.3 – *Résultat officiels de la campagne d'évaluation ESTER 2 sur les EN. La mesure de Slot Error Rate (SER) est calculée d'après la mesure de taux d'erreur mots (WER) dans les transcriptions pour les 7 participants à cette campagne. Les meilleurs résultats sont indiqués en gras.*

Les résultats présentés dans ce chapitre ont tous été obtenus sur le corpus de test d'ESTER 2, qui contient 5123 entités étiquetées manuellement. Les résultats officiels de la campagne ESTER 2 sont donnés dans la table 7.3 (lire (Galliano et al., 2009) pour une analyse détaillée). Le système présenté est décrit sous la référence **LIA**. Le résultat est indiqué en terme de *Slot Error Rate* (SER) : le meilleur système est celui qui affiche la plus basse valeur de SER. Nous observons que pour toutes les applications du système aux sorties ASR (sortie directe de transcripteur de parole), notre proposition obtient les meilleurs résultats, démontrant sa robustesse.

Nous avons également évalué le gain qui pouvait être obtenu en utilisant des méthodes de détection reposant sur l'introduction des connaissances acquises sur Wikipédia. Ceci revient à utiliser les listes de termes et leur étiquette de classe apprises selon les méthodes décrites dans le chapitre 6 pour détecter des entités en mesurant la similarité cosinus entre le contexte d'une entité et les sacs de paires mots : $TF.IDF$ d'une *métadonnée*. En ajoutant cette étape nous obtenons un gain absolu de 2% sur la tâche d'étiquetage de transcriptions manuelles, le SER passant de 28,2% à 26,2%. Le résultat final de 23,9% a été obtenu en appliquant des règles de post-traitement qui améliore la détection des limites des entités en fonction des modifications intervenues dans le guide d'annotation de la campagne ESTER 2.

7.2 Système d'étiquetage multilingue

Le système *baseline* présenté offre l'avantage de la robustesse. Son principal inconvénient est qu'il intègre très tôt dans le processus de préparation des données à étiqueter une information de nature sémantique. Nous observons dans la figure 7.1 que c'est finalement à l'étiqueteur de POS à base de HMM que revient en amont une partie du rôle d'étiqueteur des entités sémantiques relevant de la tâche d'EEN. Ceci a pour conséquence de reléguer les capacités inférentielles de l'étiqueteur CRF au second plan, alors que ce dernier pourrait théoriquement décider d'une information sémantique en se basant uniquement sur des séquences de mots et de POS. On notera d'ailleurs que dans une perspective multilingue, la plupart des étiqueteurs de POS disponibles ne

EN	AMOUNT	FONC	LOC	ORG	PERS	PROD	TIME	tous
Qté	239	196	1215	1267	1108	58	1025	5123
SER	55,82	44,64	10,74	25,04	11,66	68,45	37,80	23,91
Taux d'erreur	43,71	44,1	18,32	36,90	24,30	87,37	39,28	28,15
Fausse alarme	0,11	0,34	0,58	0,56	0,22	0,03	0,22	0,26
précision	0,85	0,61	0,77	0,79	0,93	0,53	0,91	0,86
rappel	0,56	0,559	0,81	0,63	0,75	0,12	0,60	0,718
F-Score	0,68	0,58	0,79	0,70	0,84	0,20	0,73	0,78

TABLE 7.4 – *Résultats par entité à étiqueter du système LIA appliqué au corpus ne-ref-primaire lors de la campagne ESTER 2.*

EN	AMOUNT	FONC	LOC	ORG	PERS	PROD	TIME	tous
Qté	239	196	1215	1267	1108	58	1025	5123
SER	15,27	24,90	7,09	16,08	3,63	46,03	5,850	9,80
Taux d'erreur	13,43	10,54	11,27	16,03	4,430	57,89	4,31	8,5
Fausse alarme	0,07	0,19	0,26	0,35	0,09	0,00	0,19	0,15
précision	0,93	0,818	0,897	0,89	0,97	100	0,95	0,93
rappel	0,86	0,899	0,88	0,83	0,95	0,42	0,95	0,91
F-Score	0,90	0,85	0,89	0,87	0,97	0,59	0,96	0,93

TABLE 7.5 – *Résultats par entité à étiqueter du système Xerox à base de règles appliqué au corpus ne-ref-primaire lors de la campagne ESTER 2.*

fournissent pas d'information de nature sémantique ce qui rend la méthode ci-dessus lourde à transposer. Nous proposons donc une méthode dérivée de la précédente qui supprime la phase de pré-étiquetage sémantique par HMM et la remplace par une simple phase d'étiquetage par POS d'un corpus.

7.2.1 Génération automatique de corpus d'apprentissage multilingue

Le principe est ici d'utiliser les *metadonnées* en association avec les corpus de Wikipédia pour produire automatiquement des corpus étiquetés utilisables pour entraîner une système d'EEN à base de CRF. Cette proposition part du principe que nous possédons pour chaque article encyclopédique de Wikipédia une représentation sous forme de *metadonnées*. Sachant que chaque *metadonnées* inclut dans ses paramètres une étiquette d'entité nommée, et que le corpus de Wikipédia contient de nombreux liens internes qui relient les articles encyclopédiques entre eux, il est possible d'extrapoler depuis Wikipédia en utilisant ses liens internes, un corpus étiqueté automatiquement généré.

Le processus peut être résumé par l'exemple suivant : considérons l'article *Victor Hugo*. À l'intérieur de cet article, il existe des liens internes qui dirigent vers d'autres articles encyclopédiques tels que *Panthéon*, *Besançon*, l'empereur *Napoléon III*, etc. Si nous utilisons le paramètre de classe *E.k* des *metadonnées* [5] nous savons que ces liens internes

5. Pour mémoire, *E.k* est le paramètre de la *métadonnée* contenant son étiquette de classification, par exemple PERS.HUM ; Voir le chapitre 7 pour une description de *E.k*.

correspondent à des étiquettes d'EN : *Panthéon* est un lieu (classe LOC.FAC), *Besançon* est une ville (classe LOC.ADMI) et *Napoléon III* est une personne (classe PERS.HUM). Nous pouvons donc produire une version de l'article *Victor Hugo* où chaque mot couvert par un lien interne est étiqueté par la classe correspondant à la *metadonnée* associée à l'article encyclopédique de destination.

En conséquence le fichier étiqueté généré d'après le corpus encyclopédique de Wikipédia est construit de la manière suivante. Considérons une fiche encyclopédique de Wikipédia. Nous explorons séquentiellement chaque mot de cette fiche. Nous rencontrons deux cas : un mot ou un groupe de mots est relié à un autre document encyclopédique de Wikipédia ; les mots sont des éléments textuels sans caractéristique particulière. Si un mot ou un groupe de motx est relié à un document encyclopédique cible, nous recherchons l'étiquette de classe *E.k* de la *metadonnée E* décrivant l'article encyclopédique lié, puis nous associons cette étiquette de classe en tant qu'étiquette d'EN à la séquence de mots qui supporte le lien.

Ce processus est appliqué séquentiellement sur tous les documents d'un corpus Wikipédia ce qui conduit à la construction d'un ensemble de phrases étiquetées égal à la totalité des phrases contenues dans le corpus. Cet ensemble de phrase fait ensuite l'objet de sélections qui conduisent à ne conserver finalement qu'un sous ensemble de phrases les moins bruitées possibles, directement exploitables par un classifieur CRF.

Corpus CD_1

Pour illustrer le processus (voir aussi tableau 7.6), considérons la phrase suivante extraite avec sa *syntaxe wiki* [6] depuis le corpus anglais de Wikipédia :

– *The national team of [[Republic of Kenya|Kenya]] is controlled by the [[Kenya Football Federation]].*

Comme nous savons que dans sa représentation en *metadonnée*, l'article encyclopédique de *Kenya* est associé à la classe LOC et *Kenya Football Federation* est associé à la classe ORG nous produisons le texte annoté suivant :

– *The national team of <ent begin> Kenya <ent=kenya><tag=LOC> is controlled by the <ent begin> Kenya Football Federation <ent=kenya football federation><tag=ORG>.*

Nous intitulons cette représentation C_{d1}.

Corpus C_{d2}

Nous appliquons maintenant à C_{d1} un étiqueteur morphosyntaxique [7] pour aligner le contenu textuel avec des étiquettes morphosyntaxiques. Nous avions déjà attribué

6. La syntaxe particulière utilisée pour mettre en page les documents dans le logiciel Mediawiki utilisé par Wikipédia.

7. Nous utilisons dans ce chapitre l'étiqueteur **Treetagger** en français, espagnol, et anglais. Il n'y a pas de contrainte particulière quand à l'étiqueteur, nous avons aussi mené des expériences avec **LIA_Tag**.

Format Wikipédia	C_{d1}	C_{d2} (texte)	C_{d2} (POS)	C_{d3} (EN)
The	The	The	DT	UNK
national	national	national	JJ	UNK
team	team	team	NN	UNK
of	of	of	IN	UNK
[[Republic of Kenya |	<ent begin>		*début lien loc*	
Kenya	Kenya	Kenya	NAM	LOC.ADMI
]]	<tag=loc.admi>		*fin lien loc*	
is	is	is	VBZ	UNK
controlled	controlled	controlled	VVN	UNK
by	by	by	IN	UNK
the	the	the	DT	UNK
[[	<ent begin>		*début lien org*	-
Kenya	Kenya	Kenya	NAM	ORG.DIV
Football	Football	Football	NAM	ORG.DIV
Federation	Federation	Federation	NAM	ORG.DIV
]]	<tag=org.div>		*fin lien org*	-
.	.	.	SENT	UNK

TABLE 7.6 – *Ajouts des informations successives dans les corpus C_{d1}, C_{d2} et C_{d3}.*

des étiquettes de classe d'EN. Nous obtenons C_{d2} (tableau 7.6). La représentation C_{d2} issue du contenu de Wikipédia n'est pas encore suffisante pour entraîner un système d'EEN. En effet, les liens internes de Wikipédia, ajoutés manuellement par des contributeurs, ne sont pas exhaustifs, et il subsiste de nombreuses EN potentielles non étiquetées dans le corpus. Ces absences sont susceptibles d'introduire un biais dans le processus d'entraînement des CRF.

Corpus C_{d3}

Le corpus C_{d3} va donc être produit d'après C_{d2} en rejetant les phrases contenues dans C_{d2} probablement mal ou insuffisamment étiquetées. Nous rejetons les phrases à l'aide d'une douzaine de règles. Ces règles rejettent par exemple les phrases contenant une étiquette de POS pour les noms propres qui ne sont pas associés à une EN ou encore les chiffres sans étiquettes de classes d'EN relatives à des dates ou des quantités, qui sont considérées comme *probablement* insuffisamment étiquetées. Nous obtenons finalement le corpus C_{d3} (tableau 7.6) qui contient un ensemble de phrases.

Une mesure de réduction du nombre de phrases après sélection est donnée dans le tableau 7.7. Ces phrases peuvent directement être utilisées pour entraîner le classifieur CRF.

Langage	Corpus original C_{d2}	Corpus d'entraînement final C_{d3}
Anglais	74 711 331	170 804
Français	10 008 100	167 707
Espagnol	4 900 862	152 432

TABLE 7.7 – *Quantité de phrases disponible dans le corpus d'origine et phrases subsistantes après sélection.*

Langage	Référence des EN	Étiquettes utilisées	Source
Français	ne-ref_primaire_test_ester2	PROD, ORG, PERS, LOC, FONC, TIME, AMOUNT	ESTER 2
Anglais	Corpus radiophonique	PERS, ORG, LOC, GPE	ACE phase 2
Espagnol	Magazines annotés	PROD, ORG, PERS, LOC, FONC, TIME, AMOUNT	Original

TABLE 7.8 – *Caractéristiques des corpus de test.*

7.2.2 Entraînement du CRF

L'un des intérêts de cette proposition est que l'utilisation de corpus multilingue automatiquement mis en relations avec des étiquettes d'EN en utilisant des liens internes élimine le coût de conception d'un corpus d'entraînement annoté manuellement. L'extraction de *métadonnées* depuis n'importe quelle édition linguistique de Wikipédia est la seule opération requise, et nous avons montré dans le chapitre 6 que nous disposions d'une méthode valide pour la réaliser. Wikipédia étant disponible en 272 langues [8], le processus peut être répété dès qu'un corpus étiqueté est requis dans une langue donnée. Pour démontrer cette adaptabilité nous avons mené des expériences d'entraînement et d'évaluation de système d'EEN en plusieurs langues.

7.2.3 Résultats

Langage	Précision	Rappel	F-Score	SER
Français	0,87	0,73	0,80	19
Anglais	0,90	0,83	0,86	21
Espagnol	0,89	0,81	0,84	34

TABLE 7.9 – *Résultats d'étiquetage multilingues obtenus.*

8. Source fondation Wikimedia http://meta.wikimedia.org/wiki/List_of_Wikipedias.

Nous avons généré le corpus d'entrainement de CRF C_{d3} en français, anglais, et espagnol. Nous avons ensuite appliqué le système d'EEN sur des corpus d'évaluations dans ces trois langues. Les résultats sont donnés en terme de précision, rappel, et F-mesure (qui sont les protocoles retenus pour les campagnes ESTER 2 et ACE 2). Nous avons complété cette mesure par une évaluation du *slot error rate* (SER). Le jeu d'étiquettes des étiqueteurs CRF entraînés comporte 7 classes (personnes, lieux, organisations, produits, quantités, temps et fonctions). Cette taxonomie est plus complexe que celle adoptée lors de la campagne MUC 7 et celle de la DARPA intitulée HUB 5, qui n'incluaient que 3 catégories, ou de celle de CoNLL (qui ne prend en compte que 4 classes, PER,ORG, LOC, MISC).

Nous avons évalué nos mesures en utilisant les classes de l'arbre taxonomique de la campagne ESTER 2 (PROD, ORG, PERS, LOC , FONC, TIME , AMOUNT) quand ces dernières étaient disponibles dans un corpus de test, ou des catégories équivalentes (dans le corpus ACE, par exemple, la catégorie GPE est adaptée en LOC). Nous décrivons ci-dessous les corpus de tests retenus.

Corpus de test en français

Le corpus de test français est issu de la campagne ESTER 2 et est identique à celui utilisé dans le système *baseline* décrit au début de ce chapitre. Les mesures sont réalisées sur la transcription dite de référence.

Corpus de test en anglais

Le corpus de test en anglais est celui utilisé pour la version 1.0 de la campagne d'évaluation ACE (Doddington et al., 2004). Les sources incluent des données issues de transcription de la parole dans des émissions radiophonique (comme dans le cas d'ESTER 2), télévisuelle, mais aussi des textes journalistiques. La campagne originale prévoyait des données d'expérimentation en arabe et chinois que nous n'avons pas exploité ici. Les entités sont de type personne, organisation, geo-politique, lieu, service, véhicule, et armes. L'adaptation à la norme taxonomique d'ESTER 2 est donc relativement simple (la principale différence apparente résidant dans la classe *Armes* mais qui est en réalité couverte par la classe *produits* d'ESTER 2). Nous avons utilisé pour nos mesures les transcriptions radiophoniques corrigées manuellement, semblables au corpus de référence d'ESTER 2.

Corpus de test en espagnol

Pour ce qui est de l'évaluation en espagnol, la tâche était plus délicate puisqu'il n'existe pas à notre connaissance de corpus de référence pour la tâche d'EEN dans cette langue. Nous avons réalisé une expérience simplifiée en utilisant un ensemble d'articles de presse en espagnol que nous avons étiqueté automatiquement puis corrigé manuellement.

Résultats système *baseline* appliqué dans la campagne ESTER 2

EN	AMOUNT	FONC	LOC	ORG	PERS	PROD	TIME	tous
Qté	239	196	1215	1267	1108	58	1025	5123
précision	0,85	0,61	0,77	0,79	0,93	0,53	0,91	0,86
rappel	0,56	0,559	0,81	0,63	0,75	0,12	0,60	0,718
F-Score	0,68	0,58	0,79	0,70	0,84	0,20	0,73	0,78

Résultats système multilingue

EN	AMOUNT	FONC	LOC	ORG	PERS	PROD	TIME	tous
Qté	239	196	1215	1267	1108	58	1025	5123
précision	0,90	0,98	0,77	0,92	0,92	0,32	0,97	0,87
rappel	0,74	0,45	0,89	0,60	0,93	0,35	0,62	0,73
F-Score	0,81	0,62	0,82	0,73	0,93	0,33	0,76	0,80

TABLE *7.10* – *Résultats détaillés de précision, rappel et F-Score du système multilingue français comparés au système baseline.*

7.2.4 Résultats

Les résultats obtenus par l'étiqueteur multilingue sont exprimés globalement dans la table 7.9 et détaillés pour la version française dans dans la table 7.10. On observe un niveau global de performances marquées par un F-Score légèrement supérieur à ceux obtenus lors de la campagne ESTER 2 par le système *baseline*. L'aspect intéressant est que notre système multilingue, tout en exploitant une méthode d'apprentissage entièrement automatique qui simplifie l'entraînement d'un étiqueteur dans plusieurs langues, accroît les performances globales du système. Le second aspect important est que les performances en terme de précision sont accrues et se rapprochent de celles obtenues par les meilleurs systèmes à base de règles. Ceci est essentiel dans la perspective de notre application de GAT d'après un corpus car il importe surtout que l'étiquetage des *phrases modèles* soit précis pour éviter les générations dont le sens serait erroné.

7.3 Conclusion

Nous avons présenté deux systèmes robustes d'étiquetage par entité nommées qui exploitent des *métadonnées* pour améliorer les performances d'une approche statistique. Un premier système d'EEN en français a démontré ses capacités lors de la campagne ESTER 2 et a obtenu les meilleurs résultats sur les tâches de robustesse, et la troisième place, immédiatement après des systèmes à base de règles, sur la tâche d'étiquetage d'une transcription manuelle. Pour répondre à notre objectif de génération multilingue, nous devions adapter ce système pour qu'il puisse être facilement entraîné avec des langues différentes. Il nous fallait également améliorer sa précision pour le rendre utilisable dans la perspective de modéliser des contenus de phrases qui seraient ultérieurement réemployées pour générer du texte. La proposition initiale dite *baseline*, modifiée en ce sens, a permis de valider le principe d'un apprentissage entièrement automatique pour un système d'EEN précis et performant, en utilisant les *métadonnées* décrites au chapitre 6.

Chapitre 8

Gestion et détection des co-références

Sommaire

On dit de deux ou plus expressions qu'elles sont co-référentes lorsqu'elles désignent la même entité. Les co-références et expressions référentielles sont des éléments importants du texte. Elles permettent dans un récit de fluidifier et d'alléger les phrases, en utilisant un ensemble d'expressions différentes pour exprimer un même objet sémantique. Les travaux portant sur la résolution automatique des liens de co-références sont nombreux : d'abord en raison de la complexité du problème, qui fait intervenir plusieurs niveaux d'interprétation (morphologie, syntaxe, sémantique, pragmatique). Ensuite par ce que de nombreuses applications du TAL peuvent à un moment ou un autre avoir besoin de la connaissance de telles informations. La GAT n'y échappe pas, et le travail sur les co-références est historiquement étudiée dans les conférences dédiées (INLG, ENLG notamment). On peut diviser en deux classes les algorithmes décrits dans le cadres de ces travaux : les méthodes symboliques, incluant les systèmes à base de règles, de calcul de similarité ou les systèmes de contraintes/préférence ; et les méthodes statistiques basées sur des calculs de probabilité ou par arbres de décision.

8.1 Problématique des expressions référentielles

Les expressions coréférentes peuvent être constituées par de pronoms ou des entités nommées, voire par des expressions (Jurafsky et al., 2000) [1]. Elles décrivent cinq types d'expression référentielles :

1. Les syntagmes nominaux indéfinis ;
2. Les syntagmes nominaux définis ;
3. Les pronoms ;
4. Les formes démonstratives ;
5. Les entités nommées et les noms.

Nous examinons ci-dessous ces familles d'expressions, et leurs particularités. En commençant par les **syntagmes nominaux indéfinis** qui introduisent dans le discours un élément de type entité nouveau pour le lecteur. Des exemples seraient :

- Il a cherché toute la journée pour lui envoyer *quelques fleurs*.
- Il a vu *cette magnifique Ford Falcon aujourd'hui*.

Ce sont les articles indéfinis tels que *un, une, des* (le déterminant indéfini *a* en anglais) ou les adjectifs tels que *quelques, cette*, qui introduisent la nouvelle entité et la rende identifiable.

Les syntagmes nominaux définis : les références définies introduisent une entité identifiable par le lecteur par ce qu'elle a déjà été mentionnée précédemment. Un exemple serait :

- Ceci concerne un étalon blanc que j'ai vendu à un officier. Mais le pedigree de *cet étalon* n'a pas encore été clairement établi.

Les pronoms : les références pronominales sont parmi les expressions référentielles les plus fréquentes, par ce que associées à des EN. Les méthodes pour les détecter sont donc au cœur de campagnes d'évaluation citées plus haut (telle que le GREC-NER du *Generation Challenge*). On peut rencontrer des pronoms personnels dans cette catégorie (*il, elle, lui, he, his, she, him*), les formes neutres de l'anglais (*it*) ou pronominales du français (*s'en, n'en*). Des exemples seraient :

- Emma a sourit autant qu'*elle* pouvait.
- Il a acheté une Chevrolet et a dit qu'*il s'en* était séparée.
- Jean *lui* à dit qu'*il* n'*en* voulait plus.

On observe que, pour la catégorie des pronoms, la littérature insiste sur le rôle du phénomène de *proéminence (salience)* et des contraintes qu'il induit pour la compréhension du discours. Ainsi on observe que les pronoms sont rarement introduits plus d'une

1. page 698, 21.4.1

phrase après l'entité à laquelle ils font référence. Lorsque la contrainte de proéminence est trop relâchée, des phénomènes d'ambiguïté non solubles apparaissent comme dans l'exemple de succession de phrases qui suit :

1. `John s'est rendu à la fête de Bob et a garé la Ford Falcon qu'il vient d'acheter.`
2. `Il est entré et a parlé à Bob pendant plus d'une heure.`
3. `Bob lui a dit qu'il avait choisi une voiture.`
4. `Il lui a aussi dit qu'il avait acheté la sienne hier.`
5. `Il lui a dit qu'il avait acheté la Ford Falcon hier.`

Dans cette illustration, l'entité *Ford Falcon* de la phrase 5 permet de rattacher le pronom *Il* à *John*, alors que dans la phrase 4, il n'est pas possible de rattacher *Il* à *Bob* ou *John*. On dit dans ce cas que l'entité *Ford Falcon* ne possède plus un niveau de *proéminence* suffisant pour être rattaché au pronom de la phrase 4. On mentionnera que les pronoms sont parfois utilisés en tant que cataphores, c'est-à-dire qu'ils précèdent leur référence, ce qui accentue encore leur difficulté de détection.

Formes démonstratives : les pronoms démonstratifs (*celui-ci, celle-là, ceux-ci*) peuvent apparaître sous formes de déterminants. Un exemple serait :

1. `La lavande est un fragrance.`
2. `Cette fragrance vient majoritairement de Provence.`

Les formes de surfaces des entités nommées et noms : ces formes peuvent faire l'objet de références entre elles lorsqu'elles sont exprimées sous forme de variations de graphies. Comme par exemple ici :

1. `Paris est la capitale française.`
2. `La Ville Lumière est mondialement connue.`

8.1.1 Composant de gestion de co-références dans une approche statistique

Dans une architecture classique de GAT telle que décrite dans le chapitre 1, la problématique de la gestion des co-références est plus une question de génération d'expressions de substitution (de pronoms, de noms propres et d'EN dont les graphies varient) que de détection.

En revanche, dans le cadre de notre proposition d'architecture à base de *Corpus de Phrases Modèles*, le problème est inversé : les co-références doivent être détectées en amont (c'est-à-dire lors de la conception du *Corpus de Phrases Modèles (CPM)*), à la fois pour être prise en compte lors du processus de récupération de structure de phrases existantes, mais aussi pour être administrées lors de la transformation de la *phrase modèle* choisie pour servir de support à la génération proprement dite. On pourrait illustrer cette problématique particulière par l'exemple suivant :

– `Henri Blanchet, comme Louis Dupuis, est un banquier, ce qui pour un Blanchet est une tradition familiale, ce qu'il accepte.`

Lorsque nous modéliserons les entités nommées de cette phrase, nous obtiendrons :

– `[PERS] est un banquier, comme [PERS] ce qui pour un [PERS] est une tradition familiale, ce qu'il accepte.`

Pour réutiliser cette phrase, nous aurons besoin de localiser en son sein les expressions référentielles afin d'être en mesure de substituer aux contenus de la *phrase modèle* ceux de l'*Intention de Communication*. Nous devrons donc, par un procédé de détection de co-référence, indexer les entités nommées, mais également les formes pronominales de la phrase, comme dans cet exemple :

– `[PERS.0] est un banquier, comme [PERS.1] ce qui pour un [PERS.0] est une tradition familiale, ce qu'[PERS.0] accepte.`

Ce qui nous permettra lors du processus de génération d'identifier correctement les abstractions à substituer.

Nous décrivons ci-dessous notre proposition pour résoudre cette question. Les algorithmes proposés ont été testés lors du *Generation Challenge 2010* dans la tâche GREC-NER de reconnaissance d'entités nommées et de détection d'*expressions référentielles (ER)* qui correspond très exactement à cette problématique. Pour extraire les EN et les ER, nous avons utilisé une combinaison d'étiqueteur morphosyntaxique et la méthode d'EEN présentée dans le chapitre 7.

8.2 Description du système proposé

Le système de détection qui est proposé repose sur une architecture *pipeline* dans laquelle, sont utilisées plusieurs méthodes d'étiquetage. En premier lieu, un étiqueteur morphosyntaxique est appliqué au corpus dont on veut déterminer les co-références. Dans un second temps, l'EEN détecte les entités nommées. Puis un jeu de règles logiques procède à la détection des formes pronominales. Un algorithme à base de pile FIFO[2] regroupe les expressions et entités co-référentes dans des classes. Cet algorithme est finalement complété par un détecteur de liens entre références à base de règles.

8.2.1 Étiquetage morphosyntaxique

L'étiquetage par POS est mené avec Treetagger. Les corpus étiquetés par POS sont post-traités pour associer une étiquette particulière aux prénoms (FNAME), afin d'améliorer les performances du système d'EEN. Un lexique de prénoms est utilisé à cette fin. Il en va de même pour les mots débutants par une majuscule, identifiés comme

2. First in First Out ou premier entré premier sorti.

Mots du corpus	POS	NE
Adrianne	FNAME	PERS
Calvo	NAM	PERS
enrolled	VVD	UNK
at	IN	UNK
Johnson	NAM	ORG
Wales	NAM	ORG
College	NAM	ORG

TABLE *8.1* – *Exemple de liste de mots d'une phrase avec les étiquettes morphosyntaxiques et les labels d'entités nommées.*

noms propres, et étiquetés par un label spécifique (XNAM en remplacement de NAM) (un exemple est donné dans la table 8.1, colonne *POS*). Ces étiquettes augmentent les performances du modèle de détection par CRF en améliorant les estimations de probabilités conditionnelles appliquées aux mots susceptibles de décrire une EN.

8.2.2 Entités nommées et étiquetage des pronoms co-référents

Le système EEN est celui décrit au chapitre 7 sous la référence *système multilingue*. Le système est donc capable de détecter toutes les entités nommées de la norme taxonomique de la campagne ESTER 2 (Galliano et al., 2009) (soit PERS, ORG, LOC, PROD, TIME, DATE, PERS, FONC) , bien que seules les EN de types personnes soient à détecter pour le *Generation Challenge*. On notera que la présence dans la norme taxonomique d'ESTER 2 de l'étiquette FONC (fonction professionnelle) introduit un degré de connaissance très utile pour la détection des expressions ré-férentielles de type personne. Ce système d'EEN est particulièrement performant dans le cadre spécifique du *Generation Challenge* puisqu'il obtient une précision de 0,93 sur la détection d'EN PERS. On observera que ce système est non-supervisé puisque préalablement entraîné sans recourir corpus de développement du *Generation Challenge*

A la suite du processus d'EEN les règles de détection pronominales utilisées permettent d'associer à chaque forme non étiquetée en tant qu'EN une étiquette de classe. Ces règles sont de type booléen AND et décrivent des triplets {*mot,POS, EN*}, où *word* = *{he, him, she, her ...}*, *POS tag*=NN, et *EN*=UNK. Cette structure de règles est adoptée pour éviter les détections abusives de pronoms inclus dans des noms de produits, de lieux ou d'organisation par exemple (par exemple un nom d'album musical, ou un titre de film, qui inclurait le mot *She* ou *Elle*, et préalablement associé à une étiquette de produit).

Finalement, chaque entité (EN et expression référentielle détectée) est numérotée par ordre d'apparition, et associée au numéro d'ordre de la phrase qui la contient (on notera que les mots étiquetés consécutivement reçoivent le même numéro d'index d'apparition.

8.2.3 Regroupement des entités par dépilage

L'étape finale de ce système cherche à établir la relation entre les entités co-référentes. En premier lieu, un processus de regroupement en classe est réalisé. Le principe de l'algorithme de regroupement est le suivant, les caractéristiques des entités sont indexées dans une pile (mots, POS, EN, position de la phrase dans le récit), en fonction de leur apparition chronologique dans le texte. Ainsi, l'entité située au sommet de la pile est la première détectée dans le document et est considérée comme la première entité à regrouper. Elle initialise la première classe. Cette entité initiale est comparée à toutes les autres entités présentes dans la pile. Lorsqu'une équivalence est détectée, la nouvelle entité associée est retirée de la pile et transférée dans le groupe. Quand la fin de la pile est atteinte, toutes les entités non extraites sont réordonnées, et le processus recommence avec un nouveau regroupement, en partant de la première entité actuelle de la pile. Cette opération est répétée tant que la pile n'est pas vide.

Les comparaisons des entités de la pile sont réalisées d'après deux méthodes, qui dépendent de la nature de l'entité en cours d'examen. Si nous considérons une entité potentiellement co-référente R_c de la pile P. Lors de l'itération i, le groupe courant est G_i. Chaque élément constituant l'entité co-référente R_c (par exemple *Chester FNAME Carton NAM*) est transférée dans G_i au fur et à mesure que la pile est explorée. Si $R_c \subseteq \bigcup G_i$, elle est incluse dans le groupe G_i et retirée de la pile P.

Finalement, l'inclusion des formes pronominales de $P \in R_c$ est réalisée en résolvant les anaphores, selon l'algorithme de Hobbs algorithm tel que décrit dans (Jurafsky et al., 2000) [3].

8.3 Évaluation

Cet algorithme a été évalué sur les données du *Generation Challenge 2010* organisé par le *SIGGEN* dans le cadre de la conférence INLG 2010 (Belz et Kow, 2010). Trois tâches étaient proposées lors de cette campagne. GREC-NEG, qui propose aux systèmes participants de sélectionner un groupe d'expressions co-référente au sein d'une liste. GREC-NER qui propose de reconnaître toutes les mentions de personnes dans un texte et d'identifier quelles mentions sont reliées entre elles. GREC-Full, est une tâche complète de régénération d'expressions co-référentes dans laquelle les systèmes doivent identifier toutes les mentions de personnes, et générer des expressions co-référentes exhaustives pour ces mentions.

Nous présentons dans ce chapitre les résultats obtenus sur la tâche GREC-NER, c'est-à-dire appliquée à la détection d'entités nommées et à leur regroupement. Le tableau 8.2 présente les résultats obtenus sur le corpus d'évaluation. Nous obtenons une bonne précision sur les trois corpus proposés. Notre système souffre d'une légère sous-performance sur le rappel. Nous avons observé que sur le jeu de développement intitulé *Inventors*, des problèmes de détections spécifiques étaient posés par les fortes

3. p704, 21.6.

Score	B3			CEAF			MUC		
Set	Préc.	Rappel	F-Score	Préc.	Rappel	F-Score	Préc.	Rappel	F-Score
Full set	91.48	85.89	**88,60**	85.40	85.40	**85.40**	92.15	86.95	**89.47**
Chef	91.12	87.84	**89.45**	86.53	86.53	**86.53**	91.86	88.55	**90.18**
Composers	92.01	87.14	**89.51**	86.87	86.87	**86.87**	92.11	87.02	**89.49**
Inventors	91.27	82.63	**86.74**	82.73	82.73	**82.73**	92.48	85.29	**88.74**

TABLE 8.2 – *Résultats obtenues sur le corpus de développement de la campagne GREC 2010.*

Système	Moyenne	B-3	CEAF	MUC
UDel-NER	72.71	**80.51**	**77.53**	**60.09**
Poly-co	66.99	**76.92**	**70.29**	**53.77**
LingPipe	58.23	**71.19**	**61.58**	**41.92**
OpenNLP	54.03	**67.61**	**67.61**	**33.52**

TABLE 8.3 – *Résultats obtenus sur le corpus de test lors de la campagne GREC 2010, par le système Poly-co (nom de l'implémentation présentée par l'auteur au titre du GIGL de l'École Polytechnique de Montréal, dans le cadre du projet Gitan).*

variations de formes de surfaces de certaines EN. Une solution à ce problème pourrait être d'utiliser les réseaux de formes de surfaces disponibles dans les *métadonnées* élaborées dans le chapitre 6.

8.3.1 Résultats obtenus lors de la campagne

Sans chercher à comparer des contextes d'évaluation différents (les corpus des précédentes tâches MUC étaient très éloignés de ceux exploités lors du *Generation Challenge 2010*) nous pouvons considérer que les résultats obtenus sur les jeu de tests sont bons en regard des scores habituellement obtenus par des systèmes de même nature déployés dans des campagnes précédentes. Cette première étape d'évaluation de notre système est suffisante pour intégrer un module de détection de co-références acceptable dans notre architecture d'apprentissage automatique de *Corpus de Phrases Modèles*, en vue de génération de texte. Nous avons également participé à la campagne. Les résultats officiels sont donnés dans le tableau 8.3 qui classent notre système en deuxième position, sur-performant largement les outils publics tels que LingPipe [4] et OpenNLP [5], testés dans une perspective de positionnement des systèmes présentés lors de la campagne, par rapport à l'état de l'art admis.

4. http://alias-i.com/lingpipe/
5. http://opennlp.sourceforge.net

8.3.2 Discussions

Même si les performances de notre système dans un contexte d'évaluation sont satisfaisantes et atteignent les objectifs que nous nous étions fixés d'améliorer significativement les capacités des solutions disponibles, nous observons une baisse importante des scores entre le système évalué sur le corpus de développement, et celui mesuré par les organisateurs du *Generation Challenge 2010* sur le corpus de test. La métrique MUC en particulier, mesure une baisse de performance de plus de 35 points ce qui semble très inhabituel dans de telles conditions d'expérimentation. Les observations visuelles puis statistiques que nous avons pu mener pour comparer les corpus de test et de développement du *Generation Challenge* n'ont laissé paraître aucune différence de structure qui puisse justifier un tel écart (une quantité plus marqué de pronoms dans le corpus de test par exemple, ou de plus longues distances entre co-références, auraient pu expliquer une baisse de performance de cette nature avec la métrique MUC).

Par ailleurs, les organisateurs n'ont pas été en mesure de nous fournir de copies des corpus de tests annotés - comme il est d'usage à l'issue de telles campagnes - pour que nous puissions contrôler les différents aspects de notre protocole expérimental (par exemple détecter d'éventuelles défectuosités dans nos logiciels de mesures).

Nous avons donc procédé à des tests complémentaires et rudimentaires pour vérifier les performances obtenues sur le corpus de tests en corrigeant manuellement les sorties de notre système et en comparant les sorties ainsi corrigées avec les sorties originales de notre système. Les résultats obtenus par cette méthode sont indiqués dans le tableau 8.4. Nous observons un retour à un niveau de performance proche de celles obtenues par notre système en conditions de développement. Les organisateurs du *Generation Challenge 2010* ont admis qu'un certain nombre de conversions avaient été appliquées aux fichiers des participants de GREC-NER en raisons de contraintes techniques et qu'il était possible que ces conversions aient introduit des erreurs susceptibles d'altérer les résultats finaux de l'évaluation [6].

En conséquence il nous semble acceptable de considérer que les performances réelles du système de détection de co-référence présenté dans ce chapitre, sont probablement plus proches de celles présentées dans le tableau 8.2 que de celles du tableau 8.3.

Système	Moyenne	B-3	CEAF	MUC
Poly-co	83.64	**82.90**	**81.92**	**86.12**

TABLE 8.4 – *F-Scores obtenus sur le corpus de test après ré-étiquetage par correction semi-automatique.*

6. Les échanges ont eu lieu par mails et oralement lors de la conférence INLG2010.

8.4 Conclusion

Nous avons présenté une proposition de système de détection de co-références performante, susceptible de s'intégrer avantageusement dans l'architecture d'apprentissage que nous avons conçue. Utilisé en tant que composant de notre *pipeline* de transformation d'un corpus en *Corpus de Phrases Modèles* il nous permet d'introduire dans les phrases que nous réemploierons dans l'architecture de génération de texte, une description abstraite et indexée de chaque référence présente. Ces références pourront recevoir une expression de substitution issue de *l'Intention de Communication*.

Troisième partie

Système de génération de phrases

Chapitre 9

Principes des algorithmes de génération automatique de texte

Sommaire

Dans cette partie, nous décrivons le système de *génération automatique de texte (GAT)* et ses algorithmes esquissés dans le chapitre 4. Dans notre système de GAT nous définissons une *Intention de Communication (IC)* en choisissant son formalisme. Nous appliquons un processus de *recherche d'information (RI)* qui localise un ensemble de *phrases candidates (PC)* susceptibles d'exprimer IC. Ces phrases sont issues d'un *Corpus de Phrases Modèles (CPM)* obtenu par application d'un ensemble de techniques d'étiquetages à un corpus textuel. Puis nous élaborons une mesure nous permettant de choisir dans la liste de PC, la plus adaptée pour représenter une IC donnée. La phrase choisie devient la *phrase support* de IC.

Lorsqu'il n'existe pas de *phrase candidate* correspondant à IC dans le *Corpus de Phrases Modèles*, nous appliquons une méthode de repli qui sélectionne un groupe de *phrases modèles* de type *Sujet,Verbe,Objet* (SVO), puis les agrège pour reconstruire une *phrase support*. Nous décrivons cette méthode de repli et l'agrégateur qu'elle exploite dans le chapitre 10. Lorsque la *phrase support* définitive est obtenue, c'est-à-dire une phrase identifiée comme sémantiquement compatible avec *l'Intention de Communication*, nous

appliquons un processus de transformation qui incorpore à cette *phrase support* le contenu lexical requis pour exprimer IC. Ce processus de transformation s'achève par un traitement de surface de la phrase qui rend cette dernière syntaxiquement correcte.

9.1 Algorithme NLGEN complet

Nous avons défini dans le chapitre 4 les grandes lignes d'une architecture de génération de texte exploitant la logique du premier ordre conforme au cadre de la théorie DRT. Nous avons souligné comment nous envisagions que ce mode de construction d'un générateur puisse le rendre inter-opérable avec des modèles de phrases issus d'un corpus. Nous décrivons maintenant le fonctionnement interne et les formalismes adoptés pour construire ce générateur de texte.

Considérons les données et variables suivantes :

- C est le corpus d'apprentissage utilisé pour produire le *Corpus de Phrases Modèles (CPM)* ;
- IC est une *Intention de Communication* composée d'une représentation sémantique de la phrase, d'informations lexicales qui lui sont rattachées telles que des synonymes, et des informations complémentaires telles que le temps de la phrase à générer ;
- CPM est un ensemble de phrases modélisées depuis un corpus d'apprentissage C ;
- PC est un ensemble de *m phrases candidates* susceptible de représenter sous une forme achevée IC. Toutes les PC contiennent une phrase syntaxiquement formée et, comme IC, une représentation sémantique de la phrase, des informations lexicales, et des informations complémentaires telles que le temps de la phrase à générer ;
- PS est la *phrase support* retenue pour le processus de génération ;
- PF est la phrase générée dont la forme de surface est corrigée et les éléments textuels substitués par ceux de IC.

$NLGEN$ est schématiquement représenté dans la figure 9.1 :

- $NLGEN.IC$:prépare l'intention de communication en la lexicalisant.
- $NLGEN.S$: algorithme chargé de déterminer pour IC l'ensemble PC tel que $PC \in CPM$ et $PS \in PC \rightarrow PS \sim IC$.
- $NLGEN.Sp$: si $NLGEN.S$ retourne $PC = \emptyset$, $NLGEN.Sp$ sera l'algorithme de repli activé pour reconstituer un PS d'après les éléments de IC. $NLGEN.Sp$ utilise un agrégateur de proto-phrases, $NLGEN.AGG$. Ces deux composants sont décrits dans le chapitre 10
- $NLGEN.REPLACE$: modifie les informations abstraites contenues dans PS (noms de personnes, noms de lieux, valeurs numériques) et les remplace par celles contenues dans IC.
- $NLGEN.SURFACE$: utilise un modèle de langage ML appris sur C pour corriger la surface de PS de ses éventuelles imperfections introduites par

NLGEN.REPLACE.

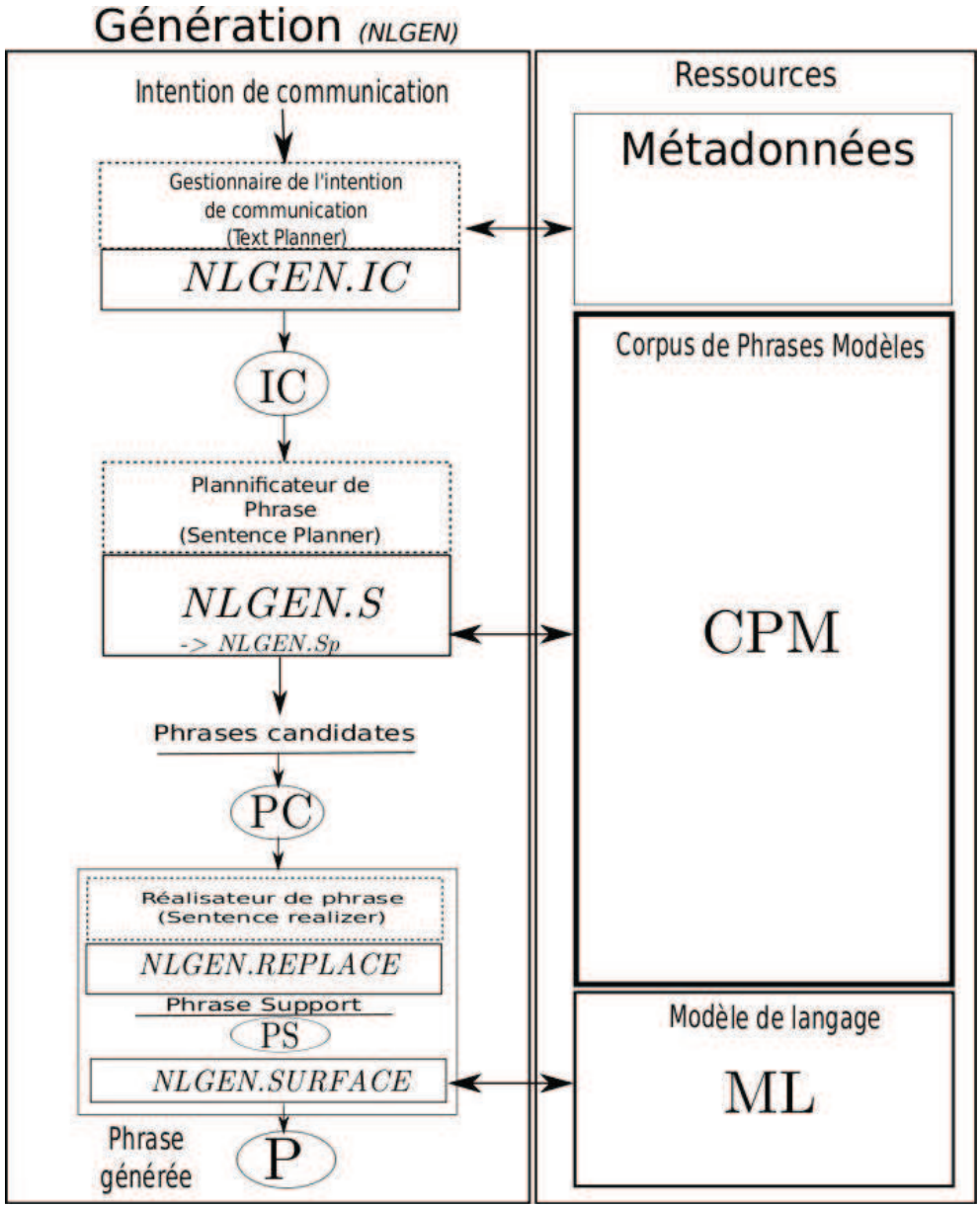

FIGURE 9.1 – *Architecture fonctionnelle de NLGEN.*

Nous allons décrire maintenant les éléments et méthodes de cet algorithme, à l'exception de l'agrégateur *NLGEN.AGG* qui par sa complexité justifie une description dans un chapitre complet (voir le chapitre 10). Par ailleurs, nous décrirons la méthode de construction des *Corpus de Phrases Modèles CPM* et CPM_{SVO} au chapitre 11. Pour les besoins de la description du fonctionnement de NLGEN, nous considérerons dans ce chapitre que l'algorithme *NLGEN* dispose toujours d'au moins une *phrase candidate PC*, qu'elle ait été déterminée par *NLGEN.S* ou produite par *NLGEN.AGG*.

Ce qui revient à la forme simplifiée suivante de $NLGEN$:

1. def=définition des besoins de communications.
2. $IC = NLGEN.IC(def)$
3. $PC = NLGEN.S(CPM, IC)$
4. $PS = NLGEN.REPLACE(PC, IC)$
5. $P = NLGEN.SURFACE(PS)$

9.2 Gestion de l'*Intention de Communication (IC)*

Le passage du formalisme de description de l'*Intention de Communication (IC)* à celui requis par le *générateur de surface*, même s'il est intellectuellement possible, n'est guère aisé à automatiser, dès que l'on dépasse le stade de la phrase élémentaire. Ceci explique assez bien pourquoi dans les systèmes adoptant l'architecture *consensuelle* telle que décrite dans le chapitre 1, on observera souvent l'introduction de composants intermédiaires, regroupés largement dans la catégorie des *composants de microplanning*.

9.2.1 IC_s : *Intention de Communication* et formalisme à logique de prédicats

Les particularités de notre système et de son formalisme à base de logique de premier ordre, nous permettent de limiter cette complexité. Si nous souhaitons générer une phrase pour indiquer que *La ville de Toulon possède son aéroport* nous pouvons définir une *Intention de Communication* avec une formule de logique du premier ordre, positive, conjonctive et existentielle : *sujet*, *objet*, *prédicat* $\Leftrightarrow$ prédicat(objet, sujet).

Elle pourrait être exprimée selon le formalisme LFG présenté dans le chapitre 1, section 1.1 sous la forme indiquée dans la figure 9.2.

$$\begin{bmatrix} pred & \begin{bmatrix} type = state \\ concept = posseder \\ temse = present \end{bmatrix} \\ arg1 & \begin{bmatrix} type = entity \\ concept = ville \\ name = Toulon \end{bmatrix} \\ arg2 & \begin{bmatrix} type = attribute \\ concept = satisfied \end{bmatrix} \end{bmatrix}$$

FIGURE 9.2 – *Représentation formelle de la phrase* ***La ville de Toulon possède un aéroport*** *selon un formalisme LFG.*

Ce qui sera équivalent à :

– $\exists$ *objet*, $\exists$ *sujet* tel que *prédicat(objet, sujet)*

Considérons la phrase IC ={La ville de Toulon possède son aéroport} qui peut être décrite comme suit en logique formelle par les variables :

$a =$ un aéroport
$t =$ une ville
$P =$ possède

Et le prédicat :

$P(t, a)$

Ce qui implicitement signifie que toute phrase du *Corpus de Phrases Modèles* dont le contenu lexical vérifie $P(t, a)$ est potentiellement candidate pour exprimer IC. Considérons maintenant une phrase s plus riche :

- s =`Jean-Baptiste Pierre Antoine de Monet est un naturaliste né le 1er Août 1744 dans le village de Bazentin-le-Petit`

qui pourrait être exprimée selon le formalisme montré plus haut avec les variables de 9.1 et les prédicats de 9.2.

$$\begin{pmatrix} e = \text{une personne} \\ l = \text{une localité} \\ m = \text{un métier} \\ d = \text{une date de naissance} \end{pmatrix} \tag{9.1}$$

$$\begin{pmatrix} A = \text{Né à} \\ V = \text{Né le} \\ M = \text{Pratique le métier de} \end{pmatrix} \tag{9.2}$$

Qui servent à construire la formule IC_s indiquée dans 9.3.

$$IC_s = \exists e \left[A(e, l) \wedge V(e, d) \wedge M(e, m) \right] \tag{9.3}$$

On remarque que IC_s vérifie l'hypothèse davidsonienne, comme supposé dans 4.2.1, tel que montré ci-dessous :

- s =`Jean-Baptiste Pierre Antoine de Monet est un naturaliste né le 1er Août 1744 dans le village de Bazentin-le-Petit.`
- s =`Le 1er Août 1744, nait dans le village de Bazentin-le-Petit le naturaliste Jean-Baptiste Pierre Antoine de Monet.`
- s =`C'est à Bazentin-le-Petit que nait en Août 1944, Jean-Baptiste Pierre Antoine de Monet, le naturaliste.`

Ainsi, à titre de justification de la souplesse de ce formalisme dans notre cadre applicatif, nous observons que nous pourrions vérifier l'existence de ces trois phrases aussi bien par une formule vérifiant l'existence d'un village ayant vu à la fois la naissance d'un naturaliste, de Jean-Baptiste Pierre Antoine de Monet, etc. C'est l'absence totale de contenu syntaxique qui caractérise cette modélisation de la phrase à produire sans pour autant en réduire le sens. Ce formalisme logique sans perte de sens rend envisageable la possibilité d'identifier un prototype de phrase dans un corpus.

On voit ici que la modélisation du contenu sémantique est beaucoup plus pratique et flexible en utilisant une logique de premier ordre, que celle gérée avec des formalismes TAG. Bos (Bos, 2008) confirme ce point de vue en déclarant pour le cadre applicatif de l'analyse sémantique que «*la logique de premier ordre semble être un bon choix lorsque la finalité est de représenter sémantiquement une phrase. Lorsque l'on souhaite associer une représentation sémantique à un fragment, par exemple des non-phrases telles que les syntagmes nominaux, les syntagmes prépositionnels, et ainsi de suite, il n'y a pas non plus d'autre choix que la logique d'ordre supérieur*».

9.2.2 IC_l : Représentation du contenu lexical

La modèle sémantique logique de l'*Intention de Communication (IC)* ne suffit pas à identifier un modèle de *phrase candidate (PC)* dans le *Corpus de Phrases Modèles*. La richesse lexicale des langues rend l'expression de contenus lexicaux multiformes par le jeu des synonymies, des métonymies, et des méronymies. À chaque modèle d'un prédicat, devra donc correspondre un espace de représentation lexical susceptible d'assurer lors de la recherche de PC l'identification de n'importe quelle abstraction candidate valable.

Pour illustrer cette idée on pourrait dire que la phrase `Jean mange une pomme` répond exactement à la même structure sémantique que `Marie mange un fruit`, est assez proche de `François déguste une poire`, et que dans les trois cas, ces phrases peuvent servir de support pour exprimer l'idée `Éric se nourrit d'une mangue`.

L'espace de représentation lexical doit donc associer à l'espace de représentation sémantique un ensemble de termes utilisables par un algorithme de *recherche d'information* (RI) pour identifier toutes les phrases lexicalement adaptables à IC. Cette représentation lexicale peut être formalisée par une abstraction : nous avons obtenu que le *Corpus de Phrases Modèles* soit déjà largement rendu abstrait par les processus d'étiquetage des entités nommées (PERS, LOC, FONC, ORG, PROD) et par celui des expressions numériques ou temporelles (DATE, AMOUNT). Ainsi, dans la représentation lexicale de IC, les noms de personnes (*Jean, Marie, François, Éric*) sont représentables universellement par l'étiquette PERS. En revanche, les notions conceptuelles verbales (*manger, naître* ...), les noms communs (*naturaliste, pomme, fruit*), les adjectifs (*petit, grand*), les locutions nominales ou verbales ainsi que les expressions et locutions (*haut-fourneau, train d'enfer*) devront être rendues plus abstraites comme dans l'exemple ci-dessous :

l_e = {PERS.HUM}
l_l = {LOC.ADMI}
l_m = {naturaliste ;scientifique}
l_d = {DATE}

On associera également un lexique de termes à chaque prédicat :

l_A = {Né à}
l_V = {Naître ;voir le jour}
l_M = {exerce}

On remarquera qu'il peut exister deux ou plus entités abstraites de la famille des entités nommées (`Jean a acheté une voiture qui ne plaît pas à Jane`). Ces formes particulières ont été différenciées lors de l'étape de détection des co-références (voir le chapitre 8) et regroupées. Il suffira donc de différencier, au sein du modèle à prédicats IC_l et de la représentation lexicale, les n entités (PERS.HUM.n ou LOC.ADMI.n) pour accorder IC_l avec les *phrases candidates* du *Corpus de Phrases Modèles*.

9.2.3 IC_c : Représentation des autres contenus

Il subsiste une dernière catégorie d'informations qui doit être identifiée. Le temps des verbes ainsi que certaines formes typiques de la nature catégorielle des phrases ou des syntagmes qu'elle contient, telles que la *déclaration, l'interrogation, l'exclamation* et la *forme exclamative*. Ces informations sont définies dans IC_c sous forme d'étiquettes de *Part Of Speech* compatibles avec l'étiquetage de POS appliqué au *Corpus de Phrases Modèles (CPM)*.

9.3 *NLGEN.S* : Algorithme de recherche d'information appliquée à la recherche de phrases candidates

Nous venons de formaliser un système de représentation à trois espaces $IC_{s,l,c}$ pour décrire IC. Chaque espace est représenté comme suit :

- Un espace de représentation sémantique modélisée par des variables, des prédicats et une logique s de IC ;
- Un espace de représentation lexical qui peut être formalisé par des vecteurs correspondants aux lexiques l de IC ;
- Un espace de représentation temporel et intentionnel qui peut être formalisé par des vecteurs correspondants aux étiquettes c de IC.

Chaque phrase contenue dans le *Corpus de Phrases Modèles* est modélisée par le système d'étiquetage en exploitant le même formalisme. On peut donc définir toute *phrase candidate (PC)*, p extraite de CPM par PC_ps, l, c. L'algorithme de recherche de PC dans

CPM utilise trois fonctions pour détecter les phrases modèles $PM \equiv IC$ (la plus appropriée pour exprimer IC). En théorie, chaque fonction est d'une importance équivalente pour le processus (en pratique l'ordre d'activation de ces fonctions joue un rôle important sur la vitesse d'exécution du système).

NLGEN.S.RTC : Recherche temporelle et catégorique

La correspondance entre l'espace de représentation temporel et celui d'une phrase doit être précise. L'un des avantages importants du système que nous proposons est la non nécessité de définir la totalité des règles de conjugaison pour générer des phrases dans une langue donnée. Ces conjugaisons sont considérées comme déjà présentes dans le *Corpus de Phrases Modèles*. Dans ces conditions, *NLGEN.S.RTC* consiste uniquement à vérifier que IC_c est strictement égal à PC_pc (que les temps des verbes d'une PC issus du corpus, identifiés par leurs étiquettes de POS sont strictement identiques à ceux prévus dans IC_c). Pour cette raison, la correspondance la fonction *NLGEN.S.RTC* retourne une valeur binaire ($0 \rightarrow$ les temps ne sont pas correspondants, $1 \rightarrow$ les temps correspondent).

NLGEN.S.RL : Recherche lexicale

L'algorithme de recherche lexicale consiste à identifier les phrase de *CPM* qui contiennent tous les éléments (verbes, noms, entités nommées) requis par *l'Intention de Communication (IC)* et contenus dans IC_l. Cette partie du système d'identification de PC doit prendre en compte la possible absence d'un ou plusieurs termes d'IC dans PC. En effet, le processus d'agrégation qui préside à la construction des phrases permet d'intégrer des éléments sémantiques de manière implicite. Pour illustrer ce phénomène, nous reprenons le prédicat :

$\exists e\,[A(e,l) \wedge V(e,d) \wedge M(e,m)]$

Qui correspond à la phrase :

`Jean-Baptiste de Monet est un naturaliste né le 1er Août 1744 dans le village de Bazentin-le-Petit.`

On observe que le verbe qui relie le lieu de naissance au sujet, identifié par la relation *né à* du prédicat *{*PERS*, né à,* LOC*}* et la représentation lexicale *{(*PERS *;Jean-Baptiste Monet), (né à ;a vu le jour), (*LOC *;Bazentin-le-Petit)}*, est en réalité implicitement exprimée par *dans*. Il n'est donc pas toujours possible de détecter la présence de l'ensemble des éléments lexicaux définis dans IC_l pour rechercher *PC*.

La correspondance entre l'espace de représentation lexical de *IC* et le contenu lexical d'une *PC* peut donc être partielle. La réponse d'une mesure de similarité doit donc être de nature proportionnelle. On adopte donc une mesure de *similarité cosinus*, $\cos(\alpha)$ qui a pour propriété de renvoyer une valeur comprise entre 0 (aucune similarité) et 1 (identicité) quelle que soit la longueur des vecteurs qu'elle compare.

Pour déployer la mesure de *similarité cosinus* on construit un lexique L ordonné par les termes utilisés dans IC_l, et ceux présents dans PC_l :

$$L = \begin{bmatrix} index_0 & = & terme_0 \\ ... & = & ... \\ index_n & = & terme_n \end{bmatrix}$$

On calcule ensuite le produit scalaire de la norme des vecteurs pour mesurer le cosinus de l'angle entre un vecteur IC_L et un vecteur PC_L de même norme que L et dans lesquels chaque ligne contient la valeur 1 si le terme d'index correspondant dans le lexique L est présent dans IC_l (pour $\vec{IC_L}$) et PC_l (pour $\vec{PC_L}$) et 0 s'il est absent. On calcule ensuite :

$$cosinus(\vec{IC_L}, \vec{PC_L}) = \frac{\mathbf{\vec{IC_L}} \cdot \mathbf{\vec{PC_L}}}{\left\|\mathbf{\vec{IC_L}}\right\| \left\|\mathbf{\vec{PC_L}}\right\|} = \frac{\sum_{i=0}^{\|L\|} IC_{Li}.PC_{Li}}{\sqrt{\sum_{i=0}^{\|L\|} IC_{Li}}.\sqrt{\sum_{i=0}^{\|L\|} PC_{Li}}} \tag{9.4}$$

On considère qu'il existe x *phrases modèles* PM dans le *Corpus de Phrases Modèles* CPM. La phrase support PS finalement retenue sera donc issue de la $PM_n \in CPM$.

NLGEN.S.RS : Recherche sémantique

On considère la formule Φ qui formalise IC_s en logique formelle. On considère l'ensemble des formules $\Phi_{0...x}$ ou $x = |CPM|$. Ce qui signifie que pour toute phrase modèle contenue dans le *Corpus de Phrases Modèles (CPM)*, il existe une formule Φ_n. Cette formule est obtenue d'après $PC_{s_n} \in CPM$. Si $\phi \equiv \Phi_n$ alors on considère que le contenu sémantique de PC peut être utilisé pour exprimer IC.

La correspondance entre l'espace de représentation sémantique et la logique sémantique d'une PC doit être précise. La réponse doit donc être de nature binaire. La formule de la PC est conforme à celle de l'*Intention de Communication* - dans ce cas, $NLGEN.S.RS = 1$ sinon $NLGEN.S.RS = 0$.

En pratique, le calcul des formules logiques est réalisé en utilisant l'algorithme présenté dans (Zouaq et Gagnon, 2010) qui consiste en une architecture *pipeline* reposant sur une analyse syntaxique de dépendances (simplifiée dans notre système par un ensemble de règles appliquées à un étiqueteur syntagmatique), suivies d'une analyse logique et d'un annotateur sémantique reposant sur une quinzaine de règles.

Calcul de score des phrases candidates

Les trois algorithmes *NLGEN.S.RTC*, *NLGEN.S.RL*, *NLGEN.S.RS* retournent une valeur qui permet d'estimer si une PC issue du *Corpus de Phrases Modèles (CPM)* est susceptible de recevoir les propriétés de l'*Intention de Communication (IC)*, le lexique de IC et la forme sémantique de IC.

On rappelle que :

- $x = NLGEN.S.RS$ et $\{x \in \mathbb{N}\} = [0, 1]$
- $y = NLGEN.S.RL$ et $\{y \in \mathbb{R}\} = [0, 1]$
- $z = NLGEN.S.RTC$ et $\{z \in \mathbb{N}\} = [0, 1]$.

Nous évaluons la similarité de p phrases contenues dans la liste PC avec IC en calculant le produit des valeurs retournés par $NLGEN.S.RS$, $NLGEN.S.RL$ et $NLGEN.S.RTC$, tel qu'indiqué dans 9.5.

$$PS = \arg \max_{PC_{r=1...p}} (PC_r x * PC_r y * PC_r z). \tag{9.5}$$

On mesure donc le degré d'adaptabilité lexicale de PC_r d'après IC par une mesure de similarité cosinus, si et seulement ci x et z sont strictement différents de 0, et c'est la PC_r de meilleure similarité qui est finalement retenue pour exprimer IC. Dit autrement, si les valeurs de x ou z - c'est-à-dire la conformité sémantique et des propriétés de PC_r avec IC sont nulles - PC_r sera rejetée pour supprimer le risque de choisir pour IC une phrase en contre-sens ou dont le temps du verbe est inadapté.

9.3.1 Algorithme de recherche ($NLGEN.S$)

En pratique, nous ordonnons les fonctions $NLGEN.S$ de pour optimiser le temps de calcul de notre système. On tient compte en particulier des contraintes logicielles suivantes :

1. Le calcul des formules de $NLGEN.S.RS$ est le plus coûteux car il implique un processus de décomposition en syntagmes et analyse des dépendances consommateur de temps de traitement. Il doit donc être appliqué en dernier à la liste des PC déjà sélectionnées.
2. La vérification de l'adéquation des propriétés (temps, négation) d'une PC à IC par $NLGEN.S.RTC$ est le processus le plus rapide puisqu'il consiste à vérifier la seule présence d'une étiquette de POS dans une PC du CPM déjà étiqueté. Par ailleurs, le retour d'une valeur nulle par $NLGEN.S.RTC$ implique la non sélection définitive de PC (voir z dans 9.3) et évite donc d'avoir à calculer $NLGEN.S.RL$ et $NLGEN.S.RS$. Cette fonction constitue donc le processus de sélection de PC le plus rapide et doit donc être appliquée en premier.
3. Le calcul de $NLGEN.S.RL$ est moins coûteux que $NLGEN.S.RS$ et permet de trier une liste de PC . On peut donc se contenter de calculer $NLGEN.S.RS$ uniquement d'après la première PC retournée par $NLGEN.S.RL$, cette fonction doit donc être activée après $NLGEN.S.RTC$ et avant $NLGEN.S$.

Ce qui donne l'algorithme suivant dont le but est de sélectionner depuis CPM des sous ensembles de PC toujours plus restreints, et qui limitent les besoins en calcul les plus importants aux plus petits ensembles :

– Algorithme $PC = NLGEN.S(CPM, IC_{rt,l,s})$.
  1. $CPM_t = NLGEN.S.RTC(CPM, IC_{rt})$ retourne dans CPM_t l'index des phrases de CPM dont les temps des verbes correspondent au temps souhaité.
  2. $CPM_l = NLGEN.S.RL(CPM_t, IC_l)$ retourne dans CPM_l un index de paires composé d'une référence de phrases dont le contenu lexical est conforme à celui du lexique de IC_l, et d'une valeur de similarité.
  3. $CPM_s = NLGEN.S.RS(CPM_l, IC_s)$ retourne dans CPM_s l'index des phrases dont le contenu sémantique est conforme à celui des prédicats de IC.
– $PC = CPM_s$

Nous diminuons ainsi autant que possible le temps de calcul requis pour évaluer une phrase avec $|CPM| <= |CPM_t| <= |CPM_l| <= |CPM_s|$ sachant que $CPM_s \subset CPM_l \subset CPM_t \subset CPM$.

9.4 Traitements finaux

Lorsque la *phrase support* (PS) est choisie dans PC, il reste à appliquer les transformations qui lui permettrons d'exprimer l'*Intention de Communication*.

NLGEN.REPLACE : Substitution des abstractions par les éléments de l'*Intention de Communication*

La fonction *NLGEN.REPLACE* remplace dans PS les entités nommées par celles décrites dans IC. Le reste du contenu lexical de PS à été sélectionné à la fois pour son adéquation sémantique et pour sa correspondance lexicale, avec IC. On localisera donc les EN (PERS, LOC, ORG, TIME, DATE) et on leur substituera les éléments fournis dans l'*Intention de Communication*.

NLGEN.SURFACE Traitement de surface par modèle de langage

Certaines erreurs syntaxiques ou grammaticales mineures peuvent être introduites par la fonction *NLGEN.REPLACE* notamment lorsque des genres, sont opposés comme dans l'exemple de transformation suivant :

– IC_l = {LOC.ADMI ;Avignon}
– PS = {`La ville de [`LOC.ADMI`] est au bord du Rhône`}
– NLGEN.REPLACE(PS, IC) = `La ville de` *`Avignon`* `est au bord du Rhône`

FIGURE 9.3 – *Exemple d'insertion d'erreurs mineures par NLG.REPLACE.*

On sait que PS est déjà correctement formée syntaxiquement et sémantiquement puisqu'extraite d'un corpus d'apprentissage considéré comme linguistiquement accept-

able. En revanche, on observe (comme dans 9.3) que des erreurs mineures peuvent être introduites.

On pourrait envisager d'appliquer la proposition de Knight décrite dans 3.2. Pourtant, bien qu'elle paraisse très proche du problème soulevé, cette dernière ne peut s'appliquer qu'à des listes de phrases sélectionnées d'après une *IC* en utilisant un modèle de langage et difficilement à une unique phrase conservée avec sa structure et réemployée.

La solution retenue consistera à apprendre un *modèle de langage (ML)* sur le corpus *C* (dont est issu le *Corpus de Phrases Modèles (CPM)*) et à appliquer le modèle de décodage pour tenter de corriger ces erreurs. On utilisera la formule du modèle de décodage appliquée à un modèle 2-gramme, comme suit :

$P(m_{c-1}) = argmax(P(m_{c-1|c})$

Ou m est un mot dans une phrase PS , $c-1$ est l'index d'un mot à vérifier, $P(m_{c-1}|c)$ est la probabilité conditionnelle du mot précédent c mesurée d'après un modèle de langage 2-gramme. On utilisera un seuil s pour déterminer à partir de quel moment, la proposition donnée par $argmax(P(m_{c-1}|c))$ doit remplacer le mot m_{c-1} originel contenu dans PS.

9.5 Conclusion

Nous avons présenté un modèle complet de génération de phrase d'après une *Intention de Communication* qui sélectionne une *phrase support* dans un ensemble de *phrases candidates* extraites depuis un *Corpus de Phrases Modèles*. Nous évaluerons les performances de ce modèle dans le chapitre 12 après avoir appliqué dans le chapitre 11 les principes d'étiquetage décrits dans la Partie II, pour constituer un *Corpus de Phrases Modèles*. Nous postulons que le modèle présenté sera de moins en moins efficace à mesure que la complexité des *intentions de communications (IC)* augmentera. Ce postulat est émis en considérant la baisse inévitable des performances de l'analyseur sémantique intégré dans la fonction $NLGEN.S.RS$ et qui est chargé de vérifier que la forme logique d'une *phrase candidate (PC)* peut à la fois recevoir les éléments *IC* et traduire le sens voulu. Pour répondre à cette difficulté, nous proposons de compléter l'algorithme $NLGEN$ par une méthode de repli. Cette méthode s'appliquera dans les cas insolubles de générations de phrases d'après des *IC* complexes et sans *PC* disponibles pour les traduire en texte. Cet algorithme de repli fragmente une *IC* en plusieurs *PC* de forme *Sujet, Verbe, Objet* et les agrège en une seule phrase générée. Il est présenté dans le chapitre 10.

Chapitre 10

Génération par agrégation de proto-phrases

Sommaire

Dans ce chapitre nous décrivons un algorithme d'agrégation de proto-phrases dont la finalité est de produire une *phrase support (PS)* lorsque la méthode de recherche de *NLGEN.S* appliquée aux *Corpus de Phrases Modèles (CPM)* n'a pas été en mesure de retourner un ensemble de *phrases candidates (PC)* correspondant à l'*Intention de Communication (IC)*. Le processus d'agrégation consiste à assembler des phrases élémentaires (les *proto-phrases*) qui chacune prise séparément correspondent à l'un des prédicats contenus dans *IC*, et ainsi à former une phrase complète neuve, correspondant exactement à l'*IC* d'origine. Notre objectif est d'utiliser la propriété des langages qui veut que la plupart des intentions de communications complexes puissent être réduites à un ensemble de structures de type *Sujet, Verbe, Objet*. Ce postulat fort repose sur des fondements théoriques solides que nous inventorions dans un premier temps. Nous commençons par rappeler les bases de la typologie des langues et leur relation à la structure *Sujet, Verbe, Objet* ainsi que la prédominance de l'ordre de ces structures. Nous mettons ensuite en perspective la forme *SVO* avec plusieurs langues et soulignons son intérêt

dans le cadre d'un système de génération multilingue à base de corpus. Puis, nous examinons les méthodes d'agrégation proposées par la littérature pour les formes SVO, observons comment ces méthodes sont implémentées dans des systèmes de GAT et terminons en décrivant notre propre implémentation d'un agrégateur SVO. Nous proposons pour finir une méthode originale d'évaluation de l'agrégateur que nous avons conçu, dans un contexte tri-lingue (français, anglais et espagnol).

10.1 Capacités multilingues du système SVO

Dans la forme SVO, l'acteur est devant, l'action de transition au milieu et le récepteur/subisseur ensuite. Ainsi, le verbe sépare clairement les 2 parties nominales de la phrase, à l'image d'un graphe :

$sujet \rightarrow relation - verbale \rightarrow objet$

Que l'on retrouve opportunément, pour l'application qui nous intéresse, dans le prédicat logique de premier ordre :

$verbe(sujet, objet)$

On notera que bien que pour les langues qui adoptent une règle générale d'ordre sujet-verbe-objet, cette typologie ne s'applique pas systématiquement, en particulier à l'oral. On trouve ainsi fréquemment dans le français quotidien des phrases qui ne suivent pas l'ordre SVO telles que :

- `Il les adore.` (ordre SOV)
- `Les bonbons, j'adore.` (ordre OSV)

À l'écrit, la transgression de la règle SVO est appréciée pour l'originalité stylistique qu'elle peut conférer à un texte, comme dans les exemples suivants :

- `Le jardinier est la plus belle rose de son jardin.` (Jean Genet) ((O)S)
- `Alors seulement Abraham le scribe se levait [...].` (Marek Alter) (OSV)

Selon Gilbert Lazard [1], l'ordre SVO *«est plus ou moins rigide selon les langues. Il est strict dans les langues sans déclinaison ou la flexion verbale (accord avec le sujet) est nulle ou réduite comme dans l'anglais, la plupart des langues scandinaves et dans une moindre mesure le français. Il est plus souple dans des langues où la flexion verbale est bien différenciée, tel l'espagnol, l'italien et l'allemand»*.

La linguistique cognitive [2] précise que sur les 6 ordres autorisés par les permutations de S, V, O, il existe trois combinaisons dominantes SVO, VSO, SOV, les trois autres étant particulièrement rares. Les trois formes dominantes sont présentes dans l'allemand et le néerlandais. L'anglais ne connaît que l'ordre SVO ; l'espagnol est majoritairement SVO (ex *«El abogado escribio la carta»*) et n'autorise le SOV que si O est

1. Gilbert Lazard, l'Études de linguistique générale :typologie grammaticale, Peeters Publishers, 2001, page 207.
2. Nicolas Delbecque, Linguistique cognitive : comprendre comment fonctionne le langage. De Boeck Université, 2006, page 28.

exprimé sous forme de pronom clictique. Le français n'admet majoritairement que les ordonnancements SVO («*je mange une pomme*») et le SOV («*il lui a fait demander*») et parfois le VSO (dans le cas interrogatif tel que «*manges tu une pomme ?*»). On citera pour le français l'existence de structures OVS rares telles que «*Bleu est le ciel*» utilisées dans certains registres (poétiques notamment).

10.2 Algorithme de génération par agrégation

La taille du *Corpus de Phrases Modèles (CPM)* et son adéquation au domaine sémantique de l'*Intention de Communication (IC)* conditionne la capacité de l'algorithme $NLGEN.S$ à retourner une ou plus *phrases candidates (PC)*. La probabilité que $|PC| = 0$ augmente à mesure que IC contient des formules de prédicats complexes ou sémantiquement distante de CPM. Si la probabilité de trouver dans un corpus de grande taille une structure *{sujet, verbe, complément}* est relativement élevée, la tâche devient de plus en plus ardue à mesure que la structure recherchée gagne en complexité. Il faut donc envisager un mécanisme de repli qui tire partie de la probabilité élevée de trouver un ensemble des phrases courtes correspondants séparément à chacune des formules logiques contenues dans IC, pour proposer malgré tout une phrase à transformer lorsque $|PC| = 0$. Si nous reprenons la phrase d'illustration vue en section 9.2.1 :

- `Jean-Baptiste de Monet est un naturaliste né le 1er Août 1744 dans le village de Bazentin-le-Petit.`

nous observons que la formule logique qui lui correspond :

$\exists e\,[A(e,l) \wedge V(e,d) \wedge M(e,m)]$

autorise également une représentation par proto-phrases telles que :

- `Jean-Baptiste de Monet est un naturaliste.`
- `Jean-Baptiste de Monet est né le 1er Août 1744.`
- `Jean-Baptiste de Monet est né dans le village de Bazentin-le-Petit.`

Même si cette suite de phrase est à la fois syntaxiquement et sémantiquement acceptable, il est évident que la répétition d'un ensemble de phrases courtes est d'une grande pauvreté stylistique. C'est d'ailleurs aussi pour répondre à ce problème de style qu'on introduit dans les systèmes de GAT classiques des mécanismes d'agrégation qui permettent de produire une phrase unique d'après plusieurs phrases courtes.

10.3 Méthodes d'aggrégation

Nous avons postulé qu'à mesure que la complexité de l'*Intention de Communication (IC)* augmente, la probabilité qu'il existe une PC dans le *Corpus de Phrases Modèles* susceptible de la représenter diminue. En revanche, il est hautement probable que

cette même IC puisse être représentée par un ensemble de phrases élémentaires (de type *prédicat(objet, sujet)*) ; cet ensemble de phrases élémentaires pourrait être à son tour transformé en une ou plusieurs phrases simples ou complexes via un processus d'agrégation. On peut donc envisager en cas d'absence de proposition de phrase complexe à l'issue du processus de RI, de développer une méthode de repli qui puisse agréger des phrases élémentaires.

10.3.1 Aggrégation des causes multiples

De telles méthodes existent dans la plupart des logiciels de GAT à minima sous forme d'assemblage de clauses multiples composées chacune de *Sujet,Verbe,Objet*. La règle d'agrégation revient dans ce cas à faire précéder la dernière phrase assemblée par un connecteur logique *ET* (*et* en français, *and* en anglais, ou *y* en espagnol). SimpleNLG, nous le verrons, intègre cette possibilité triviale dans son *réalisateur de surface*. Illustrons par un exemple avec les trois phrases :

- `Jean lit un livre.`
- `Marie regarde la télévision.`
- `Henri joue sur sa Game Boy.`

Qui peuvent être agrégées ainsi :

- `Jean lit un livre, Marie regarde la télévision et Henri joue sur sa Game Boy.`

Mais si l'agrégation de causes différentes permet de produire une phrase dont la fluidité est acceptable, il n'en va pas de même lorsque les phrases à assembler comportent des redondances. Avec la même méthode d'agrégation triviale, notre exemple plus haut donnerait :

- `Jean-Baptiste de Monet est un naturaliste, Jean-Baptiste de Monet est né le 1er Août 1744 et Jean-Baptiste de Monet est né dans le village de Bazentin-le-Petit.`

Dans ce exemple, la répétition nuit de manière importante à la fluidité et au style (même si la phrase demeure syntaxiquement et sémantiquement valide). On va donc utiliser dans ces situations des procédés d'agrégation plus sophistiqués, inspirés des méthodes rhétoriques, et dont la finalité est principalement de supprimer les répétitions.

10.3.2 Agrégation par ellipse

Il existe des processus d'agrégation qui se situent à un stade intermédiaire de la figure rhétorique, et permettent d'ajuster la phrase pour lui donner un aspect plus fluide. C'est le cas de l'ellipse. L'ellipse est une figure de style qui consiste à omettre un

Type CCE	Nom (selon ELLEIPO)	Exemples de phrases	Conditions de détection et méthode d'agrégation.
(EN)(FR) Écart (gapping)	g	Jean vit à Paris et son fils vit à Lyon / *Jean vit à Paris et son fils à Lyon*	Verbe commun.
(EN)(FR) Écart longue distance	$g(g)^+$	Mon épouse veut acheter une voiture et mon fils veut acheter une motocyclette / *Mon épouse veut acheter une voiture et mon fils une motocyclette*	Extrémités séparées par un verbe ou une locution verbale identiques.
(EN)(FR) Sous écart	$s(g)$	Le conducteur fût tué et les passagers furent sévèrement touchés / *Les conducteur fût tué et les passagers sévèrement touchés*	Conditions d'espacement et équivalence du lemme du verbe dans la section qui suit la conjonction. Temps du verbe conservé maintenu.
(EN) Entrelacement	*str*	Ma sœur vit à Berlin et ses enfants vivent à Berlin / *Ma sœur vit à Berlin et ses enfants* ***aussi***	Espace entre le sujet et l'objet occupé par un verbe de lemme identique.
(EN)(FR) Réduction de la conjonction par l'avant (FCR)	*f*	Depuis deux ans, ma sœur vit à Paris et depuis deux ans, ma sœur travaille à La Défense/ *Depuis deux ans ma sœur vit à paris et travaille à La Défense*	Identité de forme et similarité gauche (à l'intérieur des limites de la cause) des constituants principaux de la cause.
(EN) Réduction de la conjonction par l'arrière (BCR)	*b*	Anja arrived before three o'clock and Maria arrived after four o'clock. *Anja arrived before three and Maria after four o'clock*	Identité de lemme et contexte droit potentiellement en désaccord avec les contours des constituants.
(EN) Écart du sujet dans les causes avec un verbe terminal ou frontal	*s*	Into the wood went the hunter and the hunter shot a hare. *Into the wood went the hunter and shot a hare*	Sujets de formes identiques et première conjonction débutant par un verbe ou un modificateur. FCR est appliqué si autorisé.

TABLE *10.1* – *Réinterprétation pour notre NLGEN.Agg des règles de détection des ellipses rhétoriques proposées par (Harbusch et Kempen, 2009). Les règles du logiciel ELLEIPO marquées (EN) sont toutes fonctionnelles en allemand et anglais. Nous précisons dans ce tableau si la règle est utilisable en français en indiquant (FR).*

ou plusieurs éléments, en principe nécessaires à la compréhension du texte, pour produire un effet de réduction (on parle aussi *d'accélération du récit*). Il existe trois types d'ellipses [3] :

- **L'ellipse grammaticale** : elle agit par omission d'un mot ou d'un verbe. Souvent cet usage de la figure n'est pas destiné à produire un effet particulier, il s'agit avant tout de faire l'économie d'une répétition souvent par une énumération.
- **L'ellipse narrative** : elle consiste en une omission d'une séquence temporelle dans une action dramatique afin, soit d'accélérer le récit pour des raisons de commodité, soit pour dissimuler une information au lecteur ou au spectateur. L'expression «Deux semaines plus tard» révèle la présence d'une ellipse dans le récit.
- **L'ellipse poétique** : c'est une omission d'un mot ou d'un groupe de mots, parfois jusqu'à l'agrammatisme, afin de produire un effet particulier. La phrase, réduite à ses lexèmes, conserve alors son sens grâce aux intonations, comme dans les monologues intérieurs ou les impressions fugitives.

(Harbusch et Kempen, 2009) décrit un algorithme simple à implémenter et multilingue, spécifiquement dédié à l'agrégation par ellipse grammaticales (qui prennent le nom dans ce travail de *Causes Coordonnées Elliptiques (CCE)*).

La méthode proposée prend en entrée des phrases linéaires, et leur applique une ensemble de règles d'agrégation simples qui consistent principalement à fusionner les redondances. Une description des méthodes de fusion est présentée dans le tableau 10.1. Harbush et Kempen concluent qu'un module d'agrégation qui implémenterait les quatre principaux processus de 10.1 serait en mesure, avec des ajustements mineurs, de générer reproduire nombre de structures CCE pour de «*nombreux langages différents*» [4]. Cette supposition, si elle pouvait être confirmée, aurait des implications pour notre architecture de GAT. En effet, la nécessité d'intégrer dans notre architecture à base de corpus un agrégateur à règles pourrait devenir contraignante s'il fallait élaborer des règles d'agrégation pour chaque langue. Ceci aurait pour effet de restreindre sa capacité d'adaptation.

Notre propre analyse montre que seules 4 des 7 propositions du tableau 10.1 sont adaptables sans modification au français, et que par ailleurs, *str* et *b* posent un problème particulier du changement de temps du verbe. On pourrait donc en conclure que (Harbusch et Kempen, 2009), ne décrivent finalement un système d'agrégation fonctionnel sans adaptation que pour les langues qui dérivent de l'anglo-saxon. En réalité, nous montrerons qu'il est possible de construire une évolution plus générale de cet agrégateur à CCE avec très peu d'adaptation. On note que *str* peut s'adapter aux langues romanes si son implémentation prévoit la conjugaison plurielle du verbe (voir l'exemple qui donne agrégé «`Ma sœur et ses enfants vivent à Berlin`»). Nous proposons les deux heuristiques complémentaires b^{fr} et str^{fr} (voir tableau 10.2) pour résoudre ce problème.

3. Les trois descriptions sont reprises *in extenso* de `http://fr.wikipedia.org/wiki/Ellipse_(rhétorique)`.

4. NdEC : le texte original en anglais indique «*able to generate a great deal of CCE structures for many different languages*».

Type CCE	Abbr	Exemples	Conditions de détection
(FR) Réduction de la conjonction par l'arrière (BCR)	b^{fr}	Jean a écrit un article et Marie a édité deux articles/ *Jean a écrit un article et Marie **en** a édité deux*	Identité des lemmes, insertion de l'adverbe.
(FR) Entrelacement	str^{fr}	Jean lit un livre et Marie lit un livre/ *Jean lit un livre et Marie **aussi***	Suppression des syntagmes verbaux suivants le 1er et terminaison par la conjonction *aussi*.

TABLE *10.2* – *Règles de détection complémentaires proposées pour tenir compte des particularités du Français.*

10.4 Implémentation

```
TextSpec t1 = new TextSpec("my cat likes fish", "my dog likes bones",
"my horse likes grass");
Realiser r = new Realiser();
String output = r.realiseDocument(t1); //Realiser r created earlier
System.out.println(output);
```

TABLE *10.3* – *Exemple de code Java d'agrégation de causes multiples dans SimpleNLG.*

La proposition de (Harbusch et Kempen, 2009) fait l'objet d'une implémentation pour les langues allemande et néerlandaise, dans le logiciel ELLEIPO et en langue anglaise et en utilisant le langage de programmation Java dans la classe «expérimentale» [5] *ClauseAggregator* du logiciel SimpleNLG. Par défaut, SimpleNLG propose de réaliser des agrégations de phrases par détection du sujet. Il s'agit d'une implémentation partielle des méthodes d'agrégation par CCE présentées plus haut. L'une des limitations de cette implémentation est notamment qu'elle n'automatise pas le processus d'agrégation, c'est-à-dire qu'elle n'inclut pas d'algorithme susceptible de choisir la partie commune la plus favorable pour un ensemble de séquences *SVO* à assembler. Elle est par ailleurs spécifiquement conçue pour la langue anglaise ce qui imposera des modifications.

L'exemple de code Java donné dans la figure 10.3 produit le texte `My cat likes fish, my dog likes bones and my horse likes grass`. Nous pouvons conserver ce dispositif en français en remplaçant simplement la conjonction `and` par et [6]. SimplNLG devient alors en mesure d'agréger des phrases telles que celle présentée en 10.4.

5. Ce qualificatif est donné par l'auteur de SimpleNLG dans la documentation de SimpleNLG v3.8.
6. Cette modification est obtenue en remplaçant *and* par *et* dans la méthode **realiseAndList** de la classe **Realiser**.

```
TextSpec t1 = new TextSpec("Jean-Baptiste Pierre Antoine de Monet est
un naturaliste", "Jean-Baptiste Pierre Antoine de Monet est né le 1er
Août 1744", "Jean-Baptiste Pierre Antoine de Monet à vu le jour dans le
village de Bazentin-le-Petit");
```

TABLE *10.4* – *Exemple de code Java d'agrégation de phrases par ellipses dans SimpleNLG.*

Dans le cas de notre exemple plus haut, la version modifiée du module de réalisation de SimpleNLG produira la phrase : `Jean-Baptiste Pierre Antoine de Monet est un naturaliste, né le 1er Août 1744, et à vu le jour dans le village de Bazentin-le-Petit.`

```
ClauseCoordination aggregator = new ClauseCoordination();
SPhraseSpec result = new SPhraseSpec();
SPhraseSpec s1 = new SPhraseSpec("the man", "be", "hungry");
SPhraseSpec s2 = new SPhraseSpec("the man", "buy", "an apple");
result = aggregator.apply(s1, s2);
Realiser r = new Realiser();
String output = r.realiseDocument(result);
System.out.println(output);
```

TABLE *10.5* – *Exemple de code Java d'agrégation de phrases par ellipses dans SimpleNLG.*

Par ailleurs, si l'agrégateur accepte de fusionner les phrases telles que celle indiquées dans 10.5, il peut retourner un résultat tel que :

```
The man is hungry and buys an apple.
```

10.4.1 Modification et automatisation de l'agrégateur de SimpleNLG

Nous avons mis au point dans le cadre de notre module d'agrégation un algorithme susceptible, pour une *Intention de Communication* donnée, de choisir automatiquement le processus d'agrégation le plus approprié. On commencera par déterminer à quelle famille d'agrégation par ellipse peuvent correspondre un ensemble de prédicats en vérifiant dans l'ordre que :

1. Toutes les prédicats ont le même sujet ;
2. Toutes les prédicats ont le même verbe ;
3. Tous les prédicats ont le même syntagme verbal.

Si aucune des conditions précédente n'est réunie, on procède à une division de l'ensemble de prédicats jusqu'à n'obtenir que des sous groupes répondants à au moins l'un des points communs ci-dessus. Ce qui permettra d'appliquer ensuite au groupe (ou aux sous groupes) les règles d'agrégations choisies selon les formules logiques suivantes :

1. Si tous les prédicats possèdent le même verbe mais ni le même sujet, ni le même syntagme verbal, appliquer g ;
2. Sinon si tous les prédicats on le même syntagme verbal, appliquer str ;
3. Sinon si tous les prédicats ont le même sujet, mais pas le même syntagme verbal, appliquer f ;
4. En dernier ressort, on procédera à une agrégation par proposition.

Cette algorithme de détection et d'agrégation est implémenté par nos soins dans SimpleNLG, accompagné des deux versions spécifiques de b et str pour le français. Nous avons par ailleurs procédé à des modifications qui permettent de désactiver les fonctions de transformations morphologiques anglaises de SimpleNLG pour lui permettre de fonctionner avec des structures phrastiques issues d'autres langues que l'anglais et déjà formées syntaxiquement et morphologiquement. Nous vérifierons plus loin le bon fonctionnement de cette architecture pour confirmer l'affirmation des auteurs de ELLEIPO, selon laquelle quatre des principaux processus d'agrégation par ellipses suffisent pour répondre à la majorité des besoins d'agrégations. On complétera cette proposition par un nombre limité d'heuristiques afin de prendre en compte certaines typologies SVO particulières, telles que celles dont nous avons dressé l'inventaire pour le français. On notera que si ces observations se trouvaient confirmées par l'expérimentation, elles seraient alors conformes au travaux d'Haspelmath (Haspelmath, 2000)[7] qui servent de base à ceux de (Harbusch et Kempen, 2009).

10.4.2 $NLGEN.Sp$: Algorithme de recherche de phrases simples en vue de l'agrégation

Le composant d'agrégation doit ensuite s'insérer dans l'architecture *pipeline* de GAT. Pour son implémentation, considérons les éléments suivants :

- IC_{SVO} est une représentation de IC sous forme de prédicats SVO ;
- CPM_{SVO} est un sous-ensemble de CPM contenant uniquement les phrases simples de type SVO qui peuvent être extraites de CPM ;
- PC_{SVO} est un ensemble de *phrases candidates (PC)* de type SVO susceptible de représenter chaque IC_{SVO_n}.

On peut décrire le mode de fonctionnement du composant $NLGEN.AGG$ dans le *pipeline* de génération comme suit :

- $NLGEN.Sp$: si $NLGEN.S$ retourne un $PC = \emptyset$, $NLGEN.Sp$ est chargée de trouver dans CPM_{svo} les phrases qui correspondent à chaque IC_{SVO_n}. On a donc $PC_{SVO} \in CPM_{SVO}$ et $\forall n \exists PC_{SVO_n} \sim IC_{SVO_n}$;
- $NLGEN.AGG$: il s'agit de l'algorithme d'agrégation dont la finalité est de réassembler toutes les phrases SVO de PC_{SVO_n} en une seule phrase achevée PS ;
- Le résultat PC de $NLGEN.AGG$ est transmis à $NLGEN.REPLACE$ et $NLGEN.SURFACE$.

7. Tableau 2, page 40

Le fonctionnement complet de l'algorithme $NLGEN$ incluant $NLGEN.AGG$ peut se décrire comme suit :

1. $PC = NLGEN.S(CPM, IC_{rt,l,s})$.
2. Si $|PC| = 0$
   (a) $PPC = NLGEN.Sp(CPM, IC)$
   (b) $PS = NLGEN.AGG(PPC)$
3. $PS = NLGEN.REPLACE(PS, IC_{SVO})$
4. $P = NLGEN.SURFACE(PS)$

L'algorithme de recherche $NLGEN.Sp$ est équivalent à $NLGEN.S$: il cherche dans CPM_{SVO} les PC_{SVO} qui représentent chaque IC_{SVO}. Sa principale différence est qu'il va diviser IC en IC_{SVO} et procéder à une recherche itérative pour retourner dans PPC toutes les phrases qui correspondent à chaque IC_{SVO}.

10.5 Évaluation de $NLGEN.Agg$

Il n'existe pas de métrique particulière pour mesurer les performances d'un agrégateur. La littérature souligne systématiquement cette difficulté depuis une quinzaine d'années : (Radev et McKeown, 1998) [8] propose de s'en remettre à trois panels de juges humains. Dans sa description de son agrégateur, (Harbusch et Kempen, 2009) évaluent ELLEIPO en analysant 50 000 phrases en allemand, et affirme obtenir 99% de phrases conformes aux règles du logiciel sans donner aucun détails sur sa méthode [9]. Shaw teste 200 phrases dans (Shaw, 2002a) en les désagrégeant, puis en les ré-ordonnant : il indique obtenir 195 phrases correctes sur 200 selon ce processus. La thèse de Shaw (Shaw, 2002b) qui est particulièrement bien menée et exhaustive sur la question de l'agrégation, ne dédie que quelques pages à ce problème de l'évaluation.

De manière générale on peut observer que l'agrégation est considérée dans la littérature comme un moyen d'investigation de la théorie du langage plus que comme sous-fonction d'un composant tactique de la GAT dont il conviendrait de mesurer les capacités à remplir la tâche qui lui est confiée. On peut donc déduire de ces exemples qu'il est particulièrement délicat d'évaluer un agrégateur par des méthodes automatiques et qu'à ce jour aucun plan d'évaluation consensuel n'existe.

10.5.1 Méthode d'évaluation proposée

Nous proposons de tester notre implémentation sur un ensemble d'IC construite manuellement d'après des phrases existantes et obtenues par tirage aléatoire. Ces IC

8. *«we have found it very hard to perform adequat evaluation of the summaries generated by SUMMONS (qui inclut un agrégateur NdEC)»* p494.
9. section 3.4, *ELLEIPO évalué pour l'allemand*

sont ensuite soumises au module d'aggrégation. Le résultat produit est contrôlé visuellement par comparaison avec la phrase originale.

Nous extrayons aléatoirement 50 phrases pour chaque langue *SVO* depuis le corpus utilisé pour modéliser le *Corpus de Phrases Modèles (CPM)*. Dans le cadre de cette expérimentation, les 50 phrases retenues ont été extraites de Wikipédia en français, anglais et espagnol. Puis nous transformons chacune de ces 50 phrases en une suite de prédicats de la forme :

– $\exists$ *objet*, $\exists$ *sujet* tel que *prédicat(objet, sujet)*

Ceci nous donne par exemple pour la phrase *s* :

s = `Le Père Noël est un personnage légendaire lié à la fête de Noël originaire d'Europe du Nord, popularisé aux États-Unis au XIXe siècle.`

Le modèle de prédicat dont les variables sont :

p = {Le Pére noël}
f = {un personnage légendaire}
$d1$ = {lié à fête de Noël}
$d2$ = {au XIX ème siècle}
$l1$ = {d'Europe du Nord}
$l2$ = {aux États-Unis}

Et les prédicats :

E = {est}
L = {est lié}
O = {est originaire}
P = {est popularisé}

Qui nous permettent d'exprimer la phrase *s* par la formule logique :

$$\exists p\,[E(p,f) \wedge L(p,d1) \wedge O(p,l1) \wedge P(p,l2) \wedge P(p,d2)]$$

et peut être soumise en conséquence à l'agrégateur sous la forme d'un ensemble de séquences *SVO* telles que :

1. *{Le Pére noël, est , un personnage légendaire}*
2. *{Le Pére noël, est, lié à la fête de noël}*
3. *{Le Pére noël, est, originaire d'Europe du Nord}*
4. *{Le Pére noël, est, popularisé aux États-Unis}*
5. *{Le Pére noël, est, popularisé au XIX ème siècle}*

Jugement

Les 50 phrases générées par l'agrégateur d'après les structures *SVO* sont soumises à un juge humain. Ce dernier se voit présenter la phrase source *S* (celle ayant été utilisée pour bâtir une représentation décomposée en *SVO*, et la phrase cible *C* (la phrase générée par l'agrégateur d'après les *SVO*). Les phrases ainsi soumises sont évaluées par le juge humain selon les critères suivants :

1. Une note de 0 à 2 est attribuée pour décrire la **qualité syntaxique**. La note 0 étant attribué à une phrase incompréhensible du fait de sa syntaxe, 1 pour une phrase compréhensible mais incluant des erreurs, et 2 pour une phrase syntaxiquement correcte.
2. Une note de 0 à 2 est attribuée pour décrire la **qualité sémantique** c'est-à-dire le degré de restitution par *C* du sens original de *S*. La note 0 étant attribué à une phrase sémantiquement erronée, 1 pour une phrase compréhensible mais incluant des erreurs ou introduisant une ambiguïté, et 2 pour une phrase sémantiquement identique à *S*.

La jugement rendu par l'évaluateur humain suit les recommandations illustrées par les difficultés contenues dans ces deux exemples. Avec ce premier exemple :

```
S=Celle-ci protège plus de 800 espèces d' oiseaux migrateurs.
C=Celle-ci protège 800 espèces et des oiseaux migrateurs.
```

On considérera que la phrase est syntaxiquement correcte (note maximale) mais que le sens - bien que proche - n'est pas complètement restitué (note médiane). En revanche dans l'exemple suivant :

```
S=Les deux tourelles latérales sont couronnées de mâchicoulis.
C=Les deux tourelles sont latérales et couronnées de mâchicoulis.
```

Bien qu'on puisse considérer que l'esthétique de la phrase soit discutable (critère subjectif), on demandera au juge humain d'attribuer la note maximale pour ce qui est à la fois de la syntaxe et de la restitution du sens (et donc d'adopter un point de vue objectif).

Résultats d'agrégation

Les résultats de cette expérience sont indiqués dans le tableau 10.6 pour trois langues *SVO*. On observe un bon niveau de performances sur la qualité syntaxique, explicable par le principe structurel de l'agrégateur, bâti sur un système de GAT à base de règles (SimpleNLG), et qui ne permet donc pas - par nature - de produire des phrases syntaxiquement erronées dans sa langue d'origine (l'anglais). Les erreurs rencontrées en français sont dues au relâchement des contraintes du logiciel pour qu'il accepte les séquences *SVO* dans d'autres langues que celles de sa conception, sans appliquer ses règles grammaticales. On observe un maintien du sens de la phrase source dans plus

	Syntaxe			Sémantique		
Note	0	1	2	0	1	2
Langue	Erronée	Erreurs	Correcte	Erronée	Ambigu	Conforme
Français	0	4	96	5	14	81
Anglais	0	0	100	1	6	83
Espagnol	1	7	92	6	14	80

TABLE *10.6* – *Résultats obtenus par l'agrégateur pour la qualité syntaxique exprimé en score sur 100 (syntaxe à gauche, sémantique à droite).*

de 80% des cas. La plupart des ambiguïtés relevées sont de même nature que celles indiquées dans 10.5.1 ce qui signifie qu'elles affaiblissent le sens communiqué dans la phrase générée plus qu'elles ne l'altèrent, comme dans les deux exemples ci-dessous :

```
S1=Une exoplanète, ou planète extrasolaire, est une planète
qui orbite autour d'une étoile autre que le Soleil.
C1=Une planète extrasolaire est une planète et en orbite
autour d'une étoile autre que le Soleil.

S2=La Russie est le plus vaste État de la planète, situé
à cheval sur l'Europe géographique et sur l'Asie orientale.
C2=La Russie est un état, le plus vaste état de la planète,
situé à cheval sur l'Europe géographique et situé à cheval
sur l'Asie Orientale.
```

Ces affaiblissements de sens sont imputables au nombre limité de règles d'agrégation qui ne suffisent pas à traiter toutes les particularités d'une langue. Il est probable que la prise en compte de phénomènes récursifs ou de répétitions au sein d'un syntagme, permettraient de diminuer substantiellement le nombre de phrases dont le sens est affaibli. Traiter plus en avant ce problème mériterait des recherches complémentaires pour lequel le temps nous a manqué. Nous avons considéré - dans le cadre de ce travail - que ses erreurs ne constituaient pas un problème essentiel au sens ou elles n'empêchaient pas de produire des phrases syntaxiquement correcte et sémantiquement intelligibles. On peut donc envisager que le principe d'agrégation des phrases *SVO* soit acceptable, s'il est servi par un *Corpus de Phrases Modèle (CPM)* au format *SVO* de taille suffisante, activé par des IC peu complexes et servi par un algorithme de récupération des segments *SVO* à assembler performant. On peut ajouter que la méthodologie d'évaluation retenue, qui choisit de construire une IC à agréger d'après une phrase sélectionnée au hasard, si elle permet de produire une évaluation honnête, est aussi celle qui souligne le plus les difficultés du système. SimpleNLG par exemple éprouve lui même certaines difficultés pour construire une phrase telle que S1 ou S2 et impose à son utilisateur dans un tel cas un travail de conception qui minore grandement l'aspect automatisé du processus de GAT et donc son intérêt. Il conviendra néanmoins de suivre les futurs travaux qui interviendront dans le domaine de l'agrégation pour améliorer le fonctionnement du système.

10.6 Conclusion

Nous avons présenté un algorithme original de GAT exploitant une propriété simple mais performante des langues qui est l'agrégation de structures *SVO* par ellipses. Cet agrégateur par *SVO* est une alternative au problème du rôle joué par la précision de l'analyseur sémantique lors de la recherche de PM denses ou complexes. En effet, la structure des phrases *SVO* modélisées dans un *CPM* dédié est implicitement étiquetée sémantiquement : on connaît par défaut l'emplacement du Sujet, du Verbe et de l'Objet dans le segment de phrase *SVO*. On peut ainsi offrir une deuxième voie de génération par *CPM* : la charge d'extraction d'une phrase pré-existante conforme à la structure de IC peut être transférée à l'agrégateur qui va se charger de reconstruire une phrase conforme au sens contenu dans l'*IC* en partant de plusieurs séquences *SVO* plus aisées à retrouver dans un corpus. Nous voyons aussi un avantage imprévu dans l'association d'IC formalisées par prédicat de logique du premier ordre avec un agrégateur. Ce formalisme est aussi équivalent au *Resource Description Framework (RDF)*. Ce schéma est un modèle de graphe destiné à décrire de façon formelle les ressources du Web et leurs métadonnées, afin de rendre leur traitement automatique plus aisé. Développé par le W3C, RDF est le langage de base du Web sémantique. Il structure par exemple les ontologies du réseau *LinkedData*. Des ontologies telles que DBPédia (Auer et al., 2007) sont entièrement bâties d'après des triplets RDF. Cette compatibilité native entre agrégateur et web sémantique permet d'espérer qu'un système de GAT applicable à des IC construites d'après des triplets RDF, soit nativement compatible avec des contenus ontologiques du web sémantique. On pourrait donc envisager de produire un texte complet uniquement d'après la description ontologique d'une ville ou d'un pays, en extrayant ses triplets RDF représentatifs et en générant d'après eux des phrases par agrégation. Ce point ouvre des perspectives d'applications prometteuses pour la GAT.

Chapitre 11

Production d'un *Corpus de Phrases Modèles*

Sommaire

Nous avons défini les formalismes et algorithmes qui nous permettent de rechercher dans un corpus étiqueté selon les méthodes décrites dans la partie II, les phrases abstraites réutilisables dans un processus de génération. On comprend bien que l'efficacité de cet ensemble d'algorithmes est largement dépendant du corpus modélisé, qui contiendra ou non une phrase susceptible d'être transformée. Dans ce chapitre, nous étudions les notions de linguistiques de corpus et déterminons sous quelles conditions un corpus d'apprentissage sera adapté à une activité de génération. Puis nous sélectionnons un corpus C en fonction des offres numériques actuellement disponibles, et construisons deux *Corpus de Phrases Modèles (CPM)*. Le premier corpus intitulé CPM contient la modélisation d'un ensemble des phrases, obtenue d'après les méthodes présentées dans la partie II. Ce modèle est calculé sans considération envers la complexité ou la structure des phrases. Il est utilisé en tant que ressource de recherche de *phrases modèle (PM)* par l'algorithme $NLGEN.S$ décrit dans le chapitre 9. Le second CPM, intitulé CPM_{SVO} dans le chapitre 10 modélise, selon les mêmes principe que ceux appliqués pour CPM, d'après le corpus d'apprentissage C, l'ensemble des proto-phrases

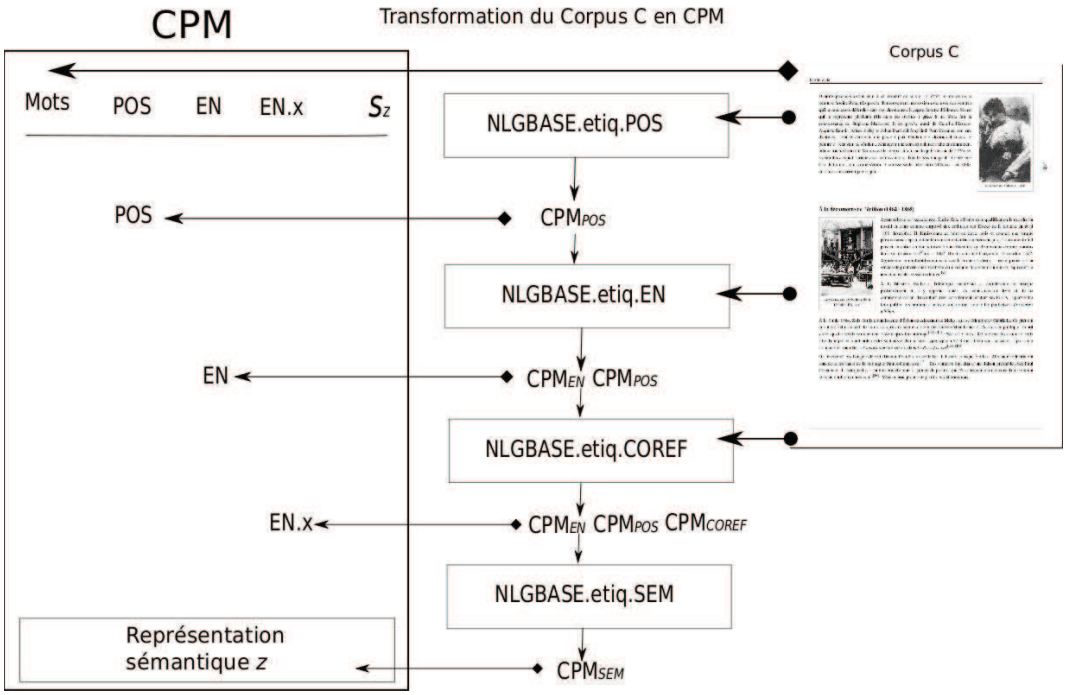

FIGURE 11.1 – *Algorithme de production de Corpus de Phrases Modèles.*

de type *Sujet, Verbe, Objet (SVO)* qui peuvent être extraites depuis C. Les proto-phrases SVO sont ensuite utilisées par l'algorithme $NLGEN.Sp$ pour identifier les composantes d'une *Intention de Communication (IC)* qui n'aurait pas pu être traitée par $NLGEN.S$. Les séquences SVO sont finalement *aggrégées* par l'algorithme $NLGEN.AGG$ décrit dans 10.

11.1 Sur le choix du corpus

Chaque individu possède un savoir linguistique, une représentation propre de sa langue. Le *Cours de Linguistique Générale* (Saussure et al., 1916), propose que [1] la langue puisse être définie par des éléments internes et externes : «*notre définition de la langue suppose que nous en écartions tout ce qui est étranger, à son système, en un mot, tout ce qu'on désigne par le terme de linguistique externe*». Pour Saussure, la *linguistique externe* touche à l'ethnologie, à la sociologie, aux civilisations, à l'histoire de cette langue, aux relations entre politique et langue, aux rapports avec les institutions. Saussure précise sa pensée [2] lorsqu'il s'exprime sur le signe et sa mutabilité : «*il faut qu'il y ait une masse parlante pour qu'il y ait une langue*». La langue existerait donc dans la collectivité «*sous la forme d'empreintes déposées dans chaque cerveau*» [3]. Une formalisation plus précise et utilisable de cette relation entre masse parlante, et langue de chaque individu interviendra en

1. (Saussure et al., 1916) page 40 Chapitre 5.
2. (Saussure et al., 1916) page 112 Principes généraux, Chapitre 2.
3. (Saussure et al., 1916) page 38 Introduction.

1986, lorsque (Chomsky, 1986)[4] proposera une distinction entre *I-Langage*, la *langue interne* et par contraste, *E-Langage* qui fait référence à un langage externe. La *langue interne* est un objet d'étude de la théorie linguistique de Chomsky ; elle se veut le modèle de la représentation mentale de la compétence linguistique qu'a un individu dans sa langue maternelle.

11.1.1 Concrétisation des rôles du *I-Langage* et du *E-Langage* dans un contexte de GAT par CPM

Cette conceptualisation d'une langue vivante et incarnée, la *langue interne* par opposition à une langue abstraite, virtuelle, en un mot culturelle, dont elle serait l'ontologie représentative, le *E-Langage* mérite attention dans le cadre d'une démarche de GAT telle que la nôtre. Elle implique en effet que seule une instanciation du *E-Langage*, sous la forme d'un *I-Langage*, peut être observable et modélisable. En d'autres termes, il ne sera pas possible d'extraire des observations générales faites sur une langue, les universaux requis pour construire les règles à déployer dans un système de production de texte.

On retrouve dans les applications de GAT à base d'apprentissage cette distinction entre langue abstraite externe, culturelle et sociale, et langue réelle, c'est-à-dire instanciée et régulée par un groupe plus ou moins important d'individus : elle implique que l'entraînement du système de génération doit tenir de compte du domaine couvert par la représentation instanciée d'une langue dans un corpus pour s'assurer qu'une proposition de phrase existera dans le contexte applicatif du destinataire de la communication textuelle. Concrètement, ceci signifiera que l'on aura besoin d'un sous-ensemble descriptif d'une langue propre aux avocats pour écrire aux avocats, propre aux journalistes pour écrire aux journalistes, et ainsi de suite.

Nous avons choisi de travailler à la fois sur des intentions de communication et des corpus d'apprentissage de nature encyclopédique. Nous utilisons pour l'apprentissage et la construction de corpus d'évaluations la ressource Wikipédia qui offre l'avantage de fournir de vastes quantités de phrases dans de nombreuses langues. Mais on devra garder à l'esprit que ce choix n'est pas sans conséquences : le fait que plus de 25% de Wikipédia contienne des biographies de personnes[5] et moins de 12% de description de produits (elles mêmes très disparates puisqu'elles décrivent aussi bien des avions que des boissons sucrées) implique qu'on trouvera nécessairement plus de *PM* pour générer des phrases biographiques que des phrases descriptives de produits.

Nous insistons donc sur le fait que toute implémentation de notre proposition de GAT devra prendre en compte le domaine couvert par le corpus d'entraînement (son I-Domaine) en regard du domaine de communication désiré. C'est une particularité de notre système.

4. Chapitre 2 *Concepts of languages*.
5. Voir les statistiques sur http://www.nlgbase.org/fr/stat/stat_clust.html.

11.2 Génération des *Corpus de Phrases Modèles (CPM)*

Le *Corpus de Phrases Modèles (CPM)* est une représentation d'un corpus textuel C qui inclut des informations de nature morphosyntaxique (le temps des verbes, la natures des mots), conceptuelles abstraites [6] telles que les entités nommées, les quantifications, les dates, les informations co-référentielles (la nature d'un pronom par exemple), et pour finir, la forme logique de la phrase c'est-à-dire sa structure représentée par un prédicat de la logique du premier ordre. La forme schématisée d'une *Phrase modèle (PM)* est décrite dans la figure 11.1. Le processus de construction de CPM d'après C peut être mené à bien avec l'*architecture pipeline* que nous décrivons maintenant.

- $NLGBASE.etiq.POS$: qui est un étiqueteur morphosyntaxique. Dans notre application, nous utilisons Treetagger qui est choisi pour ses capacités multilingues.
- $NLGBASE.etiq.EN$: qui est un étiqueteur d'entités nommées (EEN). Nous l'avons décrit dans le chapitre 7. Il fonctionne avec un classifieur CRF et utilise en entrée un mot et une étiquette de POS.
- $NLGBASE.etiq.COREF$: qui est un étiqueteur de co-références, décrit dans le chapitre 8. Il repose sur une analyse des mots, des étiquettes de POS et EEN.
- $NLGBASE.etiq.SEM$: qui est un étiqueteur sémantique décrit dans 5.4.

Les entrées et sorties du pipeline seront agencées comme suit :

1. $CPM_{POS} = NLGBASE.etiq.POS(C)$
2. $CPM_{EN} = NLGBASE.etiq.EN(C, CPM_{POS})$
3. $CPM_{COREF} = NLGBASE.etiq.COREF(C, CPM_{POS}, CPM_{EN})$:
4. $CPM_{SEM} = NLGBASE.etiq.SEM(C, CPM_{POS}, CPM_{EN}, CPM_{COREF})$:

Ce qui nous permet d'obtenir CPM tel que pour chaque ligne x de CPM nous avons : $\forall x CPM_x = \{C_x, CPM_{POS_x}, CPM_{EN_x}, CPM_{COREF_x}\}$. On peut présenter CPM comme une suite de ligne composées d'un mot, de son étiquette morphosyntaxique, d'une étiquette d'EN (UNK étant l'étiquette attribuée aux mots qui ne sont pas des EN), et une étiquette d'EN éventuellement attribuée aux co-références. Le CPM est divisé en n segments qui correspondent à n phrases s complètes, et pour tout s nous avons une représentation sémantique CPM_{SEM_s} qui lui est associée. Un exemple de segment de CPM est donné dans la table 11.1. CPM_{SEM_s} dans ce tableau correspond à la définition sémantique $CPM_{SEM_{s=0}}$, définie par les variables :

$p0$ =Henri,$p1$ =Ravaillac

Par les prédicats :

A =assassine , D =décédé

Et la formule :

$\exists p_0 A(p_1, p_0) \wedge D(p_0)$

6. On rencontre aussi dans la littérature l'expression «formes sérialisées».

x	C	CPM_{POS}	$CPM_{EN.COREF}$	CPM_{SEM}
0	Il	PRO :PER	PERS.0	$s.0.p0$
1	est	VER :pre	UNK	$s.0$
2	parti	VER :pper	UNK	$s.0.D(p0)$
3	,	PUN	UNK	$s.0$
4	Henri	FNAM	PERS.0	$s.0$
5	,	PUN	UNK	$s.0$
6	assassiné	NOM	UNK	$s.0.A(p1, p0)$
7	par	PRP	UNK	$s.0$
8	Ravaillac	NAM	PERS.1	$s.0.p1$
9	.	SENT	UNK	$s.0$

TABLE *11.1* – *Niveaux d'étiquetage disponibles dans CPM.*

11.3 Structure du *Corpus de Phrases Modèles (CPM)*

Pour les besoins de nos expériences les CPM sont produits d'après les corpus français, anglais, et espagnols de Wikipédia. La quantité de phrases étiquetées obtenues pour chaque corpus est indiquée dans le tableau 11.2. Ce nombre de phrase correspond à ce qui peut être obtenu après application d'un ensemble de traitement de préparation au corpus original : retrait de la syntaxe Wiki utilisée par le logiciel MédiaWiki, suppression des phrases mal formées (par exemple d'une longueur exagérément longue), suppression des éléments particuliers telles que les URL ou les références bibliographiques.

Langage	Corpus originalC_{d2}
Anglais	74 711 331
Français	10 008 100
Espagnol	4 900 862

TABLE *11.2* – *Quantité de phrases étiquetables obtenues d'après trois éditions linguistiques de Wikipédia.*

Nous présentons dans les tableaux 11.2, 11.3 et 11.4 plusieurs mesures réalisées sur les phrases du corpus Wikipédia en français (WP-FR). La courbe 11.2 indique le pourcentage de phrases comportant une conjonction pour un groupe de phrases de longueur donnée. On observe ainsi que sur 836 926 phrases de 3 mots, soit 8,36% du nombre de phrases totales contenues dans le corpus, seules 2 351 contiennent une conjonction (soit 0,28%). En revanche, sur 160 445 phrases de 23 mots, 85 207 contiennent au moins une conjonctions (soit 53,10%). L'anomalie dans la distribution, marquée par le nombre de phrases à 4 mots contenant une conjonction, soit 108 677 phrases avec conjonctions sur un total de 634 729 phrases de 4 mots (17,12%), vient d'une particularité de Wikipédia qui intègre de nombreux titres avec connecteur logique tels que : *«Liens et documents externes»*.

Ces mesures nous permettent d'évaluer la proportion de phrases complexes avec connecteurs logiques au sein du corpus. Cette information est importante car le traite-

ment d'une phrase complexe (qui peut être facilement réduite d'après ses constituants séparés par des connecteurs logiques) est différent de celui d'une phrase simple[7].

Dans Wikipédia, une phrase de plus de 40 mots non complexe sera par exemple :

```
Le 2 janvier 1956, lors des élections législatives, après sa traversée
du désert, il retrouve son mandat de député du 6e secteur de la
Seine (Aubervilliers, Saint-Denis, Montreuil, Vincennes), sous
l'étiquette Front républicain, avec une confortable avance.
```

Des phrase de 45 mots, complexes, seront par exemple :

```
La redécouverte par l'Occident des vertus thérapeutiques du cannabis
est généralement attribuée à Sir William Brooke O'Shaughnessy,
qui en 1831 publie dans la revue médicale britannique The Lancet
sa méthode d'injection intraveineuse d'électrolytes en solution
pour soigner le choléra.

Rodeo (Chili )( Sport ): Le rodéo est un sport où un cavalier
monté à cheval, le huaso, doit faire pression sur un autre animal,
généralement un jeune taureau, contre le mur d'une enceinte fermée.
```

Une phrase de 45 mots, complexe, avec une connexion logique, sera par exemple :

```
La Déesse tient dans la main droite un boute-feu, servant à allumer
la mèche des canons, [et] montre de la gauche l'inscription du
socle: la réponse du maire de Lille, André, refusant la reddition
de la ville assiégée.
```

Le graphique de la figure 11.2 décrit la distribution des phrases en fonction de leur longueur. Le profil de courbe en longue traîne est caractéristiques de la distribution de pareto. Il est donc intéressant d'observer que la distribution de la longueur des phrases en fonction du nombre de mots qu'elle contient suit une loi proche de celle de Zipf qui approche la description du rapport entre fréquence et rareté des mots. On compte 838 000 phrases de 3 mots dans WP-FR et 980 phrases de 96 mots. La répartition devient non significative à partir de 136 mots (les nombres de phrases se stabilisent alors entre 10 et 100 pour chaque échantillon).

La courbe de la figure 11.4 présente sur les abscisses la distribution du pourcentage de conjonctions contenues dans les groupes de phrases de n mots, indiquées sur les ordonnées. Cette courbe décrit la complexité des phrases en fonction de leur longueur. On observe que plus les phrases sont longues, plus elles sont fréquemment complexes. Ainsi sur 836 926 phrases de 3 mots, seules 2351 sont avec conjonctions (0,28%). On note que plus aucune phrase n'est non complexe à partir de 50 mots de longueur.

7. On retient ici pour la qualification de phrase complexe toute phrase composée de plusieurs propositions. Ce qui revient presque toujours à ce qu'elle inclue plus d'un verbe conjugué.

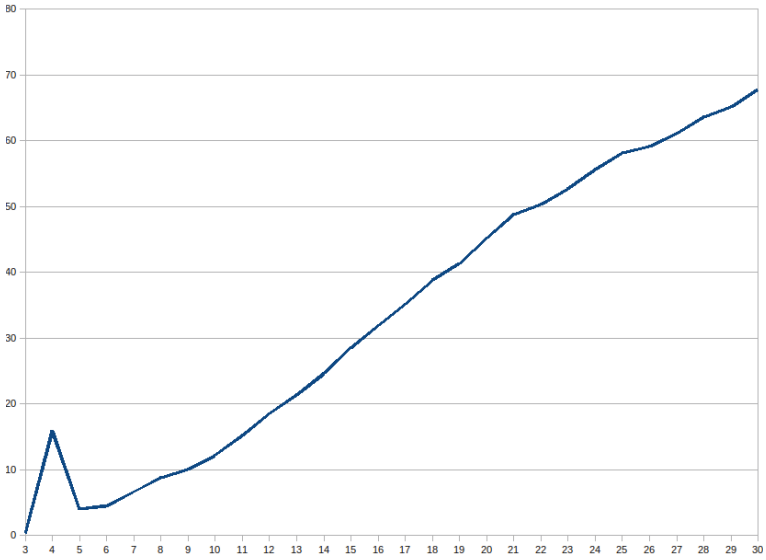

FIGURE *11.2* – *Distribution de phrases en fonction de leur longueur dans le corpus Wikipédia.*

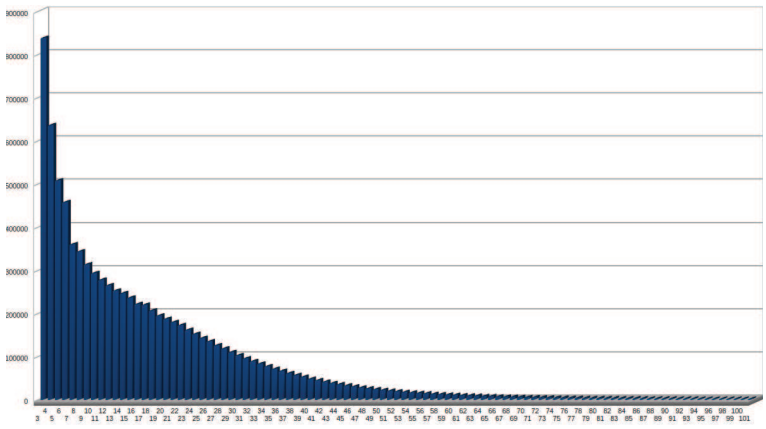

FIGURE *11.3* – *Distribution de phrases non complexes en fonction de leur longueur dans le corpus Wikipédia.*

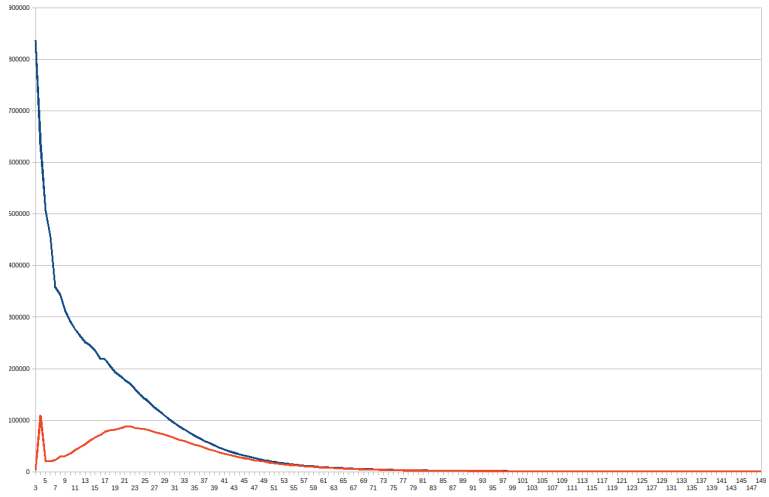

FIGURE *11.4* – *Distribution de phrases selon le pourcentage de conjonctions présentes dans des groupes de phrases de n mots, dans le corpus Wikipédia.*

Nombre de verbes	Nombre de phrases de *n* verbes
1 verbe	1763010
2 verbes	1263851
3 verbes	876742
4 verbes	576251
5 verbes	363278
6 verbes	218948
7 verbes	129663
8 verbes	77279
9 à 19 verbes	45637 à 710

TABLE *11.3* – *Proportion de phrases de n verbes dans le corpus Wikipédia FR.*

11.3.1 Structures et répartition des verbes dans *CPM*

Le nombre de verbes contenus dans une phrase détermine son degré de complexité et conditionne aussi son admissibilité au ré-emploi dans notre architecture de GAT. Nous souhaitons que le plus grand nombre de phrases possibles utilisent un petit nombre de verbes, à la fois pour identifier rapidement et efficacement des structures *SVO* de qualité et des *phrases modèles* aisément analysables sémantiquement. Le tableau 11.3 montre la structure verbale des phrases de WP-FR. On retrouve des proportions similaires dans les principales langues disponibles pour cette encyclopédie. Près de 2 millions de phrases de WP-FR contiennent 3 verbes ou moins. C'est une quantité suffisamment importante pour envisager de supporter des IC relativement complexes et nous limiterons ultérieurement les *IC* de nos expériences à cette structure.

11.4 Génération du *Corpus de Phrases Modèles (CPM)* limitées aux SVO

Le processus d'agrégation décrit par l'algorithme $NLGEN.AGG$ dans le chapitre 10 demande de décomposer une IC en plusieurs proto-phrases IC_{SVO} afin de les agréger dans un second temps. Pour ce processus spécifique, nous produisons un second CPM dédié intitulé CPM_{SVO} et entièrement composé de représentations étiquetées de phrases de type SVO. Pour y parvenir, nous récupérerons dans C toutes les phrases ou les morceaux de phrases de type SVO qui composeront CPM_{SVO}. Par morceaux de phrases, nous indiquons que nous ne nous restreignons pas uniquement aux phrases simples et SVO déja présentes dans le corpus : une telle restriction serait extrêmement limitative si nous considérons que les phrases SVO sont rares (moins de 10% du corpus) ; se restreindre à ces phrases existantes aurait pour conséquence de produire un CPM_{SVO} avec peu de propositions, et difficile à exploiter pour $NLGEN.Sp$. Nous donnons ci-dessous des exemples de ces phrases non SVO qui peuvent être morcelées et donc qui peuvent être transformées en phrases SVO :

- **Il évolue en première division** du championnat du Queensland.
- **Le Kouchk est un cours d'eau** très irrégulier.
- **Candijay est une municipalité** de la province de Bohol.

On peut procéder également avec les phrases complexes à n verbes, comme :

- Le Jersiais **Charles Robin fonde un poste de pêche** à Chéticamp en 1780 mais **le village est habité** de façon permanente à partir de 1782, lorsque **des réfugiés de la Déportation des Acadiens viennent y vivre**.

11.4.1 Étiquetage des syntagmes

Pour générer CPM_{SVO}, nous commençons par identifier les structures syntagmatiques de C en syntagmes en utilisant l'étiqueteur Treetagger. Ce dernier offre l'avantage de prendre en charge toutes les langues que nous proposons d'intégrer à notre système (anglais, français, espagnol), mais à pour inconvénient sa lenteur d'exécution [8]. Il manque par ailleurs parfois de fiabilité dans son processus d'étiquetage. Nous postulons que ce manque de fiabilité, générateur de bruit lors de la décomposition en proto-phrase, est compensé par l'abondance de séquences SVO obtenues.

11.4.2 Méthode à base de règles de combinaisons syntagmatiques

Nous avons d'abord tenté de sélectionner les SVO par les syntagmes en essayant un ensemble de règles. Cette méthode est très difficile à mettre au point car elle se

8. La décomposition de 10 millions de phrases occupe environs 120 heures/processeur (T6600 2,2GHz).

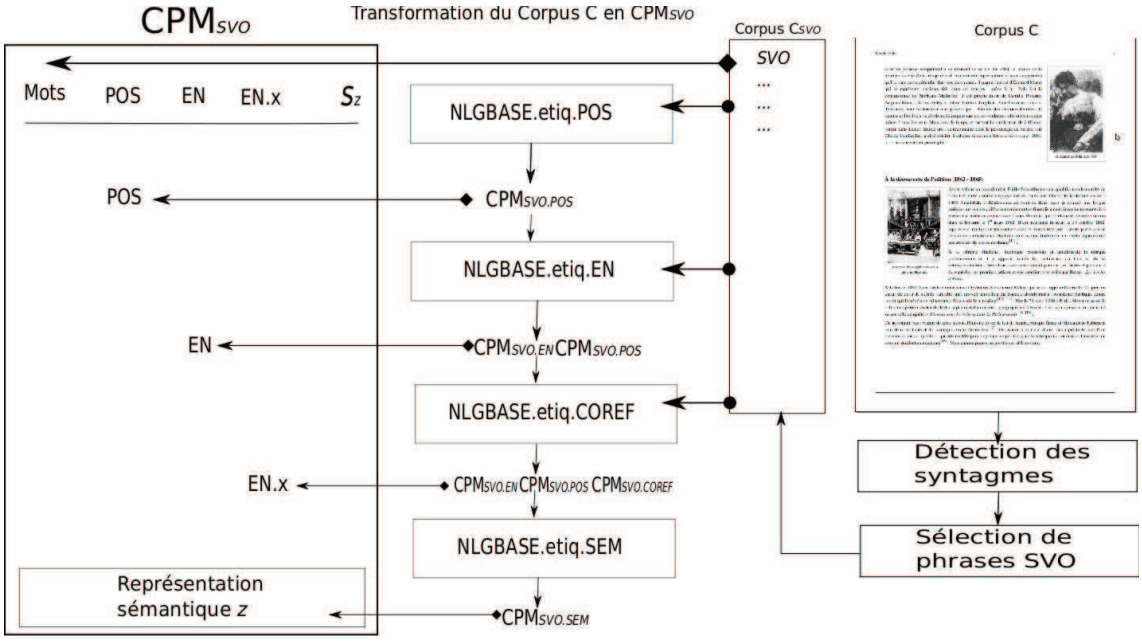

FIGURE *11.5* – *Algorithme de production de corpus SVO.*

Séquence	Exemple
S NP VN AP	elle est fermée
S NP VN PP	cette page présente les éléments
S VN PP	il remonte dans le passé
S NP VN{avoir} NP NP	Le vieux Biff avait prévenu son double
S NP VN{à.VER :pper} VN NP	Marty réussit à échapper à Biff

TABLE *11.4* – *Formes des règles adoptées pour extraire des structures SVO.*

heurte à la complexité des combinaisons syntagmatiques. Les règles indiquées dans le tableau 11.4, trop restrictives, revenaient à rejeter plus de 70% de phrases *SVO* utiles. Le deuxième inconvénient de cette méthode est qu'elle est particulièrement dépendante de la langue et qu'elle impose finalement une mise au point assez complexe de chaque version linguistique du système de GAT. Nous avons donc abandonné l'idée de nous en remettre exclusivement aux syntagmes et avons porté nos efforts sur une méthode alternative.

11.4.3 Extraction de structures *SVO*

Cette deuxième méthode d'extraction repose toujours sur l'étiquetage des syntagmes, mais est conjuguée à un ensemble de règles de sélection heuristiques utilisables pour les trois langues de destination. Elle divise les phrases selon les critères indiqués dans le tableau 11.5. Les morceaux de phrases obtenus d'après le découpage proposé dans ce

Abr	Séquence appliquée sur *C*	Application
p	Division des phrases par la ponctuation	, ; ! ? :
m	Division des phrases par marqueur de début et de fin de phrase	$< s >$, ?, !
c	Division des phrases par les connecteurs logiques et coordinations étiquetées	et, aussi, <COORD>
s	Division des phrases par les conjonction de subordination	afin | lorsque | puisque | comme | si

TABLE *11.5* – *Règles de segmentation de phrases.*

tableau sont ensuite analysés et rejetés ou conservés en fonction des syntagmes qu'ils contiennent. En grammaire, une phrase peut être considérée comme un ensemble autonome, réunissant des unités syntaxiques organisées selon différents réseaux de relations plus ou moins complexes appelés subordination, coordination ou juxtaposition. Nous tirons partie de ces propriétés en utilisant dans un second temps les règles du tableau 11.6. Dès que la sélection est achevée, nous obtenons une variante de *C* contenant uniquement des phrases *SVO* que nous intitulons C_{SVO}. Nous appliquons le même processus de transformation de *C* en *CPM* pour transformer C_{SVO} en CPM_{SVO}, à savoir le pipeline d'étiquetage :

1. $CPM_{SVO.POS} = NLGBASE.etiq.POS(C_{SVO})$
2. $CPM_{SVO.EN} = NLGBASE.etiq.EN(C_{SVO}, CPM_{SVO.POS})$
3. $CPM_{SVO.COREF} = NLGBASE.etiq.COREF(C_{SVO}, CPM_{SVO.POS}, CPM_{SVO.EN})$:
4. $CPM_{SVO.SEM} = NLGBASE.etiq.SEM(C_{SVO}, CPM_{SVO.POS}, CPM_{SVO.EN}, CPM_{SVO.COREF})$

On notera que dans le cadre *SVO*, la phase d'analyse sémantique :

$NLGBASE.etiq.SEM(C_{SVO}, CPM_{SVO.POS}, CPM_{SVO.EN}, CPM_{SVO.COREF})$

est triviale puisqu'elle revient à identifier le sujet à gauche, le verbe au milieu, identifié par son étiquette morphosyntaxique, le complément étant la chaîne subsistant à droite. L'un des avantages ici est donc que l'étape d'analyse sémantique, si cruciale et délicate dans le cas de l'utilisation de *NLGEN.S* et *CPM* devient quasiment anecdotique. La contrepartie de cette trivialité est qu'elle fait reposer tout le poids de la qualité finale de la phrase produite sur l'agrégateur *NLGEN.AGG*.

11.4.4 Nombre de phrases extraites

Le tableau 11.7 présente les résultats produits par cette méthode. On remarque qu'il est possible de collecter un peu plus de 30% de structures *SVO* sur le total de *phrases modèles (PM)* proposées par le corpus *CPM* ; que les segments analysés représentent au total un peu plus de une fois et demi le total de phrases disponibles (16 millions de segments traités pour 10 millions de phrases dans le *C* français par exemple) ce qui signifie que chaque phrase de *C* permet en moyenne d'examiner 1,6 segment. Les

Abr	Séquence appliquée sur C	exemple accepté	exemple rejeté
PNV	Débutant par un noyau verbal ne sont conservées que si ce noyau inclut un pronom.	<VN> il retrouve </VN><NP> Sehorn </NP>	<VN> est </VN> <NP> Ya-Ya </NP> <NP> qui </NP> <VN> sort </VN> <PP> sur <NP> le label Fury </NP> </PP>
SNV	Les morceaux de phrases sans noyau verbal sont rejetées.		<NP> le premier week-end </NP> <PP> de <NP> <NP> septembre
+NV	Les morceaux de phrases avec plus de 1 noyau verbal (phrases qui demeurent complexes malgré la division) sont rejetées.		<NP> Une </NP><PP> des <NP><NP> stratégies </NP></PP><AP> qu </AP> <AP> '<VN> il adopte </VN> <VN> <VN> consiste à </VN> <VN> <VN> plonger à </VN> <NP> <NP> environ 50 m </NP> <PP> de <NP><NP> profondeur
cW	Limite par le nombre de mots n avec une limite c	si $n < c$ conservation (et application des autres règles)	si $n > c$ rejet

TABLE *11.6* – *Règles de sélection par acceptation ou rejet appliquées après segmentation.*

Langue	Nombre de phrases dans C	Segments de phrases de C rejetées	Segments de phrases SVO contenus dans CPM_{SVO}
Français	9 991 431	12 901 363	2 940 000
Anglais	74 711 331	91 060 100	24 000 051
Espagnol	4 900 862	6 800 711	1 210 420

TABLE *11.7* – *Segments SVO générés d'après les Corpus de Phrases Modèles complets.*

segments de phrases rejetées sont constitués principalement de structures rhétoriques incomplètes (par exemple dans *Jean a voyagé à New-York **avec son frère***, la section **avec son frère**, qui résulte d'une agrégation, n'est pas récupérable et est rejetée).

11.5 Conclusion

Nous avons présenté deux méthodes de construction de *Corpus de Phrases Modèles (CPM)* qui exploitent les méthodes d'étiquetage décrites dans la partie II. En utilisant un corpus encyclopédique de grande taille, nous avons pu produire dans trois langues un CPM composé de *phrases modèles (PM)* plus ou moins complexes, et un CPM_{SVO} in-

tégralement composé de phrases de type *Sujet,Verbe,Objet* susceptibles d'être agrégées. Nous avons évalué comment, lorsque la complexité des phrases contenues CPM augmentait, la probabilité de rareté des *PC* candidates pour une *Intention de Communication (IC)* donnée devenait importante. Nous avons ainsi confirmé la nécessité d'utiliser une solution alternative par agrégation pour être en mesure de générer une phrase quelle que soit la complexité d'une *IC*. Dans le prochain chapitre, nous mesurons les performances des algorithmes de génération de notre système, lorsqu'ils sont associés aux CPM que nous venons de produire.

Chapitre 12

Évaluation du système NLGEN

Sommaire

L'évaluation des systèmes de GAT est un sujet controversé et central de la discipline. Il est évident que la nature même de la tâche pose un problème logique - en apparence insoluble - en particulier pour la question de l'évaluation automatique. Élaborer une métrique de définition qualitative d'un texte produit par une machine implique que l'on ait à sa disposition une connaissance pré-existante ou une méthodologie d'observation de ce qui caractérise la qualité d'un texte. Cette méthodologie voudrait que ce qui définit son sens, son intelligibilité, voire son esthétique soit formalisée et théorisée. Évidemment, aucune théorie linguistique valide et démontrée ne décrit de manière sûre, reproductible et calculable, ce qu'est la bonne construction syntaxique ou le rapport entre le sens voulu par le producteur du texte et son interprétation par son destinataire. Si une telle connaissance existait, le problème de la GAT (mais pas seulement lui) serait résolu. Cette même question - cruciale - se retrouve d'ailleurs dans les tâches périphériques de la GAT telle que la *Traduction Automatique (TA)*, le *Résumé Automatique (RA)* et même, dans une moindre mesure, de *l'Analyse sémantique (AS)*. Il est d'ailleurs symptomatique que l'AS se dégage peu à peu de cette difficulté en adoptant les formalismes logiques proches de la DRT, évidemment plus simples à mesurer. Finalement, la problématique de l'évaluation de la GAT demeure un problème largement ouvert qui pourrait même expliquer selon certains auteurs, tel (Scott et Moore, 2007), la relative discrétion de la discipline au sein des sciences du langage.

12.1 La difficile évaluation des systèmes de GAT

De nombreuses propositions ont été faites au cours des vingt dernières années pour accompagner les développements des processus génératif du TAL tels que la traduction, le paraphrasage, le résumé et la GAT. On peut considérer que trois grandes familles de méthodes d'évaluation s'appliquent à la GAT :

1. Les méthodes humaines, c'est-à-dire par observation des phrases produites et notation selon d'un ensemble de critères par des juges.
2. Les méthodes basées sur l'accomplissement d'une tâche.
3. Les méthodes automatiques à base de métriques.

Une investigation particulièrement poussée a été menée récemment dans (Reiter et Belz, 2009) sur la problématique comparée de l'évaluation des systèmes de GAT selon ces trois méthodes. Les auteurs concluent que les métriques associées à des méthodes automatiques possèdent des «*propriétés attractives*» [1]. Elles sont considérées comme rapides, simples, et reproductibles. Leur impact dans plusieurs domaines du TAL leur confère également une «*crédibilité indéniable*», selon ces auteurs.

Les métriques automatisées suivent une évolution constante. (Saggion et al., 2002) proposaient par exemple d'utiliser la similarité cosinus associée à la correspondance d'unigrammes et la longueur des sous-séquences communes pour mesurer la qualité d'un résumé automatique d'après un texte de référence. Deux métriques - BLEU et ROUGE - sont particulièrement étudiées en raison de leur proximité avec la problématique de la GAT, à savoir le besoin de vérifier qu'un texte généré est à la fois sémantiquement et syntaxiquement correcte. Ces métriques sont très utilisées dans les activités proches que sont la TA, le RA et la GAT.

12.1.1 L'influence des métriques *BLEU* et *ROUGE*

Le principe de BLEU (Papineni et al., 2002) est d'utiliser une forme modifiée du score de précision pour comparer une traduction obtenue avec un système de TA avec plusieurs traductions de référence. La métrique BLEU fournit une mesure allant de 0 à 1, ses auteurs prenant soin d'indiquer que peu de traductions peuvent obtenir un score de 1 puisque ce dernier signifierait que le texte évalué serait exactement identique à une des traductions de référence. En revanche plus les phrases de référence sont nombreuses dans la traduction, plus le score de BLEU doit se rapprocher de 1. A titre d'exemple, il est précisé (Papineni et al., 2002) que sur un test de 500 phrases, un traducteur humain obtient un score de 0,34 sur quatre références, et 0,25 sur deux références. BLEU consiste à comparer les n-grammes d'une *phrase candidate* avec les n-grammes de la traduction de référence, et à compter les séquences identiques sans tenir compte de leur position. La précision est ensuite calculée en divisant le nombre de n-grammes identiques par le nombre de mots de la traduction candidate.

1. «*Automatic evaluation metrics have many desirable properties*», in (Reiter et Belz, 2009), page 555.

ROUGE (Lin, 2004) est orientée vers le résumé automatique. La méthode consiste en un calcul de rappel sur des *n*-grammes entre un résumé produit et un ensembles de résumés de référence. ROUGE est déclinée en plusieurs variantes : *n*- grammes (ROUGE-N) ; sous-chaînes les plus communes (ROUGE-L et ROUGE-W), dans une version qui utilise une fenêtre glissante de *n* bigrammes (ROUGE-S) et ROUGE-SU qui rectifie un biais de ROUGE-S. ROUGE-S (S pour *Skip-bigram*) mesure statistiquement le recouvrement des co-occurrences entre un résumé candidat et un résumé de référence. ROUGE-S n'attribue pas de score à un résumé candidat dont aucune paire de mot n'est co-occurrente avec les références (si par exemple la phrase candidate est l'exacte inverse de la référence telle *Bleu est le ciel* $\rightarrow$ *Le ciel est bleu*). ROUGE-SU ré-introduit un comptage d'unigrammes pour compenser ce biais.

Par leur mode de fonctionnement, ces méthodes paraissent bien adaptées à l'évaluation de systèmes de GAT. Il suffirait de concevoir un ensemble de textes de références ou d'extraire des *intentions de communications* d'après ces textes de référence pour ensuite générer un texte et mesurer sa qualité d'après BLEU ou ROUGE. C'est la voie que choisissent la plupart des auteurs de systèmes de GAT qui mettent ces mesures en œuvre. On peut pourtant leur opposer que tant BLEU - orienté phrases - et ROUGE - plus adapté à la mesure de cohérence d'un texte complet, sont mal adaptés à l'évaluation du texte qu'un composant de surface est généralement amené à produire, essentiellement en raison de la déconnexion de ce dernier avec un document de référence (une traduction ou un résumé modèle).

12.1.2 Adaptation des métriques à la GAT

À de nombreuses reprises, des aspects de systèmes de GAT (Langkilde-Geary, 2002; Habash, 2004) ont été testés en s'aidant de BLEU. Mais les auteurs sont généralement unanimes (Reiter et Belz, 2009; Scott et Moore, 2007; Callison-Burch et al., 2006)(Knight) [2], pour affirmer que les méthodes automatiques sont très discutables pour évaluer les contenus textuels générés et encore moins adaptées pour mesurer la qualité opérationnelle des systèmes de GAT.

En comparant les scores produits par BLEU et ROUGE avec des évaluations humaines appliquées au domaine de génération du bulletin météorologique, les résultats décrits par (Reiter et Belz, 2009) suggèrent que l'utilisation de méthodes automatiques peut être appropriée dans le cadre de la GAT, avec un certain nombre de précautions. Ces méthodes seraient utilisables notamment pour évaluer la qualité linguistique des textes générés mais en complément d'une évaluation humaine plutôt qu'en remplacement.

Finalement, nombreuses sont les prises de positions récentes en faveur d'évaluations exclusivement humaines : ainsi (Koller et al., 2009) propose une évaluation par le web et d'autres, plus classiquement, une méthodologie rigoureuse faisant intervenir des juges (Reiter et Belz, 2009).

2. Dans sa conférence en tant qu'invité à EACL 2006, cité par (Reiter et Belz, 2009)

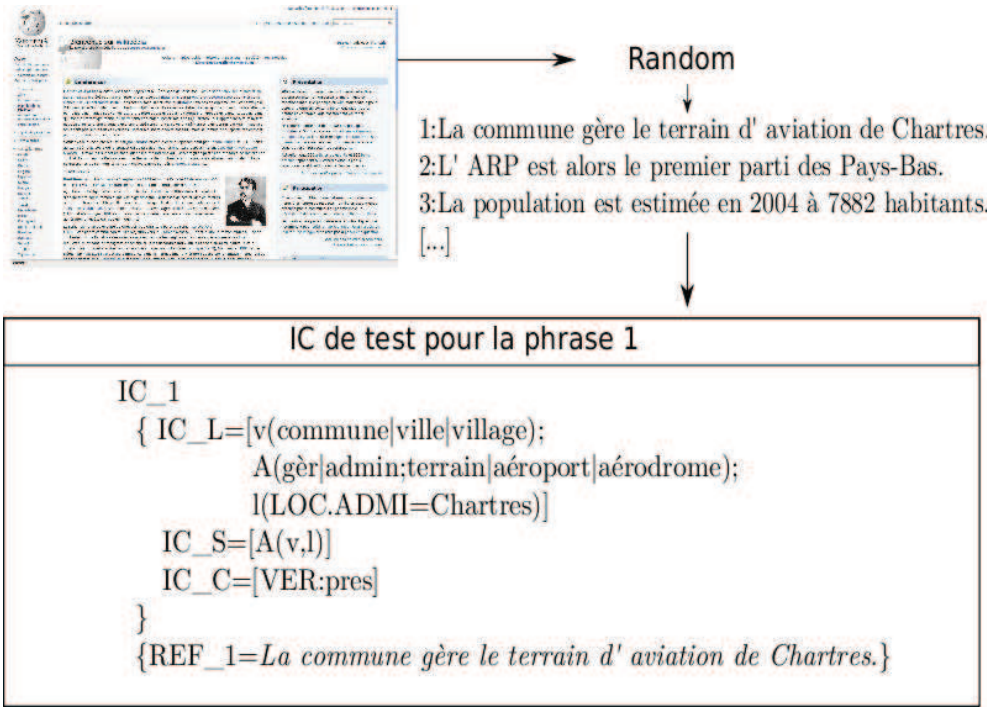

FIGURE 12.1 – *Construction des IC d'évaluation d'après des phrases tirées aléatoirement.*

12.1.3 Méthode d'évaluation retenue

Dans le cadre particulier du présent travail, nous nous sommes posé la question de l'utilisation des méthodes BLEU et ROUGE pour évaluer les textes générés par notre système. Plusieurs difficultés qui nous sont spécifiques sont introduites par ces métriques : en premier lieu, nous n'avons aucun corpus textuel de référence à leur soumettre pour les configurer (selon le principe des traductions de références de BLEU). En second lieu, l'idée de mesurer la cohérence des phrases produites d'après des *n*-grammes n'a que peu de sens lorsque nous cherchons justement à ré-utiliser la créativité d'un humain en reprenant à l'identique la structure des phrases qu'il a déjà écrites pour véhiculer une IC. Pour finir, l'une des particularités essentielles de notre système de GAT est qu'il repose pour une large part sur des méthodes de *Recherche d'Information (RI)*. C'est donc autant le potentiel du générateur à produire une phrase accordée à l'IC que nous voulons mesurer, que sa capacité à retrouver dans un *Corpus de Phrases Modèles (CPM)* la *phrase support (PS)* la plus appropriée. Or, dans les méthodes existantes d'évaluation citées plus haut, la problématique de la RI - très spécifique à notre système - n'est jamais envisagée.

Toutes ces remarques nous ont conduit à adopter une méthode d'évaluation classique et humaine. Elle nous paraît être la plus appropriée dans notre contexte de GAT (en particulier selon les arguments convaincants développés par (Reiter et Belz, 2009)). Nous avons élaboré un protocole et un corpus expérimental inspiré du *Penman exercise*

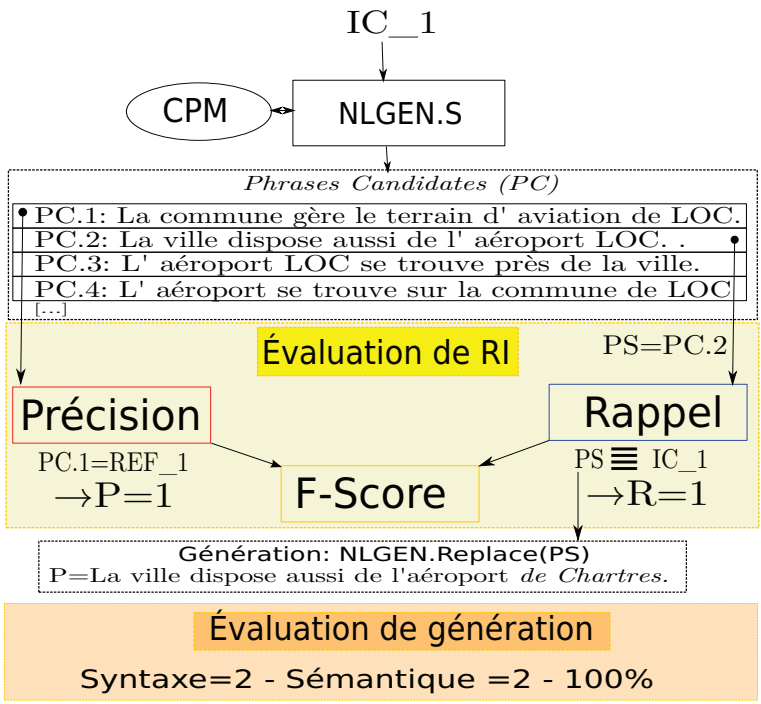

FIGURE 12.2 – *Méthode de mesure de performances des IC d'évaluation.*

set, conçu pour tester le générateur KPML anglais [3] :

1. Dans un premier temps, nous extrayons aléatoirement n phrases d'un corpus (voir figure 12.1) que nous appelons *Phrases d'Origine (PO)*.
2. Puis nous construisons une IC_n pour chacune de ces n phrases *PO*, selon les normes définies dans le chapitre 9 (voir figure 12.1).
3. Nous soumettons ces IC aux deux générateurs de texte (correspondants à *NLGEN.s* et *NLGEN.sp* selon le chapitre 10) (voir figure 12.2).
4. Nous évaluons la qualité de la phrase générée en la comparant à l'original ayant servie à élaborer IC en utilisant les règles d'évaluation proposées dans 10.5.1 (voir figure 12.2).

Caractéristiques de l'évaluation

Nous nous sommes posé la question de l'intérêt d'une comparaison des sorties de notre système avec d'autres, comme cela se fait en RA ou en TA. Il serait possible, par exemple, d'utiliser les IC du *Penman exercise set* cité plus haut pour tenter de générer une phrase avec notre système. Le *Penman exercise set* consiste, comme dans notre proposition, en une représentation formelle de chaque IC accompagnée de sa phrase source et

3. Voir http://www.fb10.uni-bremen.de/anglistik/langpro/kpml/genbank/R3b12-English/Docu/ENGLISH-nigel-exerciseset-mismatches-19981209/index.html

de celle générée. Il existe, aussi des descriptions formelles adaptées à KPML pour le système HealthDoc [4].

On pourrait appliquer ces descriptions formelles d'IC à notre système et à SimpleNLG, pour établir des comparaisons. Mais une telle évaluation comparative est très difficile à mettre au point pour plusieurs raisons. En premier lieu, ces échantillons sont sémantiquement très éloignés des contenus du corpus Wikipédia que nous avons utilisé pour l'apprentissage de notre système. Ils sont liés à des corpus spécialisés tel Reuter pour les *Penman exercise set*, ou à un domaine applicatif très particulier (le bulletin météorologique, le rapport de médecine). Il présentent donc une différence de nature sémantique qui hypothèque l'intérêt de la comparaison. Cette dernière revenant finalement à évaluer l'incapacité de sortir de son domaine sémantique d'exercice. Par ailleurs ils sont tous proposés en anglais et ne sont donc d'aucune utilité pour comparer des capacités multilingues. Pour finir, ils sont surtout conçus pour tester la capacité de production syntaxique d'un système à base de règles de grammaire. Or nous voulons évaluer plusieurs aspects très spécifiques à notre système, non centrés sur la problématique de la syntaxe et de la surface des phrases :

1. La capacité du système à localiser une *phrase modèle (PM)* dans le *Corpus de Phrases Modèles (CPM)* quelle que soit la complexité de IC.
2. La capacité du système à localiser n séquence SVO dans le *Corpus de Phrases Modèles SVO* (CPM_{SVO}) pour reconstruire par agrégation une IC.
3. La qualité sémantique de la phrase finalement générée pour une IC donnée.

Pour répondre à ces besoins particuliers et compte tenu des difficultés évoquées plus haut, nous avons donc opté pour une évaluation qualitative et non comparative. Nous ne faisons en cela que suivre la méthodologie habituelle d'évaluation des systèmes de GAT, qui n'ont à notre connaissance jamais été évalués de manière comparative dans des campagnes normatives, comme cela a pu être fait en TA et en RA.

Pour chaque aspect du système (méthode avec PM complète obtenue quelle que soit l'IC, et méthode par agrégation de IC_{SVO}) on veut vérifier que la méthode de RI sait retrouver, si elle existe, une phrase qui exprime parfaitement IC. On utilise une mesure de précision et de rappel adaptée. Prévue pour l'évaluation d'une multi-classe, cette mesure est exposée dans sa forme canonique dans les formules de calcul 12.1 et 12.2.

$$Rappel_i = \frac{\text{nombre de documents correctement attribués à la classe i}}{\text{nombre de documents appartenants à la classe i}} \tag{12.1}$$

$$Précision_i = \frac{\text{nombre de documents correctement attribués à la classe i}}{\text{nombre de documents attribués à la classe i}} \tag{12.2}$$

4. Voir http://www.fb10.uni-bremen.de/anglistik/langpro/kpml/genbank/R3b12-English/Docu/index.htm

On veut mesurer la qualité du processus de RI dans notre contexte applicatif en vérifiant à la fois sa capacité à retrouver une phrase et que cette phrase soit sémantiquement adaptable à IC. On adapte pour ce motif le calcul de précision et rappel selon les formules 12.3 et 12.4.

$$Rappel_{IC} = \frac{nombre\ de\ PS = PO}{|IC|} \tag{12.3}$$

$$Précision_{IC} = \frac{nombre\ de\ PS \equiv IC(\text{vérifiées par un juge humain})}{|PS|(\text{considérées correctes par le système})} \tag{12.4}$$

$$F - Score = \frac{2 \cdot (\text{Précision}_{\text{IC}} \cdot \text{Rappel}_{\text{IC}})}{(\text{Précision}_{\text{IC}} + \text{Rappel}_{\text{IC}})} \tag{12.5}$$

La formulation de 12.3 et 12.4 est inhabituelle. L'idée est, avec le rappel, de vérifier que l'algorithme de RI est en mesure de retourner en premier rang exactement la même phrase que celle ayant servi à construire IC en étant passé par toutes les étapes d'analyses sémantiques, lexicales et morpho-syntaxiques. Ceci valide le principe de la formule décrite dans le chapitre 9.5. Ensuite, c'est avec la précision que l'on mesure la capacité du système à retourner une PS, différente de PO, mais qui puisse néanmoins exprimer correctement une IC. On obtient donc avec cette méthode une mesure à la fois des performances de la RI (par le rappel) et des capacités de la formule d'évaluation de similarité lexicale, morpho-syntaxique, et sémantique décrite en 9.5 (par la précision). Le calcul du rappel est automatique (égalité entre PS et PO). Celui de la précision implique un contrôle humain (vérification visuelle que PS choisi au rang 2 de $PC \equiv IC$, voir illustration 12.1). On évalue pour finir le F-Score classiquement par 12.5.

Pour toutes les PS retournées que l'on considère comme capables d'exprimer IC on active le processus de génération, qui produit la phrase finale P, ce qui permet de mesurer la conformité sémantique et syntaxique finale de la phrase produite selon les critères suivants :

1. Une note de 0 à 2 est attribuée pour décrire la **qualité syntaxique**. La note 0 étant attribué à une phrase incompréhensible du fait de sa syntaxe, 1 pour une phrase compréhensible mais incluant des erreurs et 2 pour une phrase syntaxiquement correcte.
2. Une note de 0 à 2 est attribuée pour décrire la **qualité sémantique** c'est-à-dire le degré de restitution par P du sens original de IC. La note 0 étant attribué à une phrase sémantiquement erronée, 1 pour une phrase compréhensible mais incluant des erreurs ou introduisant une ambiguïté, et 2 pour une phrase sémantiquement identique à IC.

12.2 Résultats

L'évaluation est menée avec 125 *IC* construites d'après 50 phrases extraites aléatoirement de Wikipédia en français, 50 en anglais et 25 en espagnol.

12.2.1 Évaluation de la recherche de phrases support

Les résultats de recherche de *phrases support (PS)* dans le CPM sont indiqués dans les tableaux 12.1 et 12.2.

La méthode de collecte de *PS* complexes et adéquates pour exprimer une IC riche est logiquement la moins performante (tableau 12.1) : elle est à la fois dépendante de la rareté de phrases à la fois sophistiquées et conformes à une IC donnée, mais aussi de la qualité de l'analyseur sémantique utilisé. La précision (maximum de 0,91) est bonne et signifie que la formule de sélection de PS dans PC fonctionne correctement. La différence de performances entre l'anglais et les autres langues est due à l'utilisation d'un analyseur sémantique (Johansson et Nugues, 2008) particulièrement performant. La relativement faible quantité de phrases retournées conformes à IC (rappel de 0,60 soit à peine plus de la moitié des IC satisfaites) indique que dans 40% des cas, soit le CPM ne possède pas deux phrases (la phrase d'origine, tirée aléatoirement, et celle de second rang issue de PC) capables de véhiculer la même *IC* soit l'analyseur sémantique a mal interprété les structures des *PC* contenues dans CPM. On observe que le volume du CPM issu de Wikipédia en anglais permet de disposer d'une quantité de *PM* plus importante que dans l'édition française, ce qui explique les meilleurs performances de rappel.

Langue	Rappel	Précision	F-Score	*IC*
Français	82	54	65.11	50
Anglais	91	61	73,03	50
Espagnol	71	43	53,56	25

TABLE 12.1 – *Résultats de recherche obtenus par NLGEN.S sur la totalité des IC.*

Les résultats des systèmes de recherche de *PS* de type *SVO* sont indiqués dans le tableau 12.2. On observe de très bonnes performances sur la récupération de structures *SVO* ce qui est cohérent : ces séquences sont simples, relativement fréquentes et la probabilité qu'une séquence *Sujet, Verbe, Objet* correspondant aux éléments d'une IC_{SVO} soit présente dans des corpus de la taille de Wikipédia est forte.

Discussion

L'amélioration des performances de *NLGEN.S* (tableau 12.1) est conditionnée par la précision de l'analyseur sémantique. Un analyseur sémantique qui serait plus précis dans ses analyses des *phrases candidates (PC)* du CPM, permettrait d'augmenter le

Langue	Rappel	Précision	F-Score	IC_{SVO}
Français	91	92	90,02	148
Anglais	92	94	91,97	162
Espagnol	81	84	82,47	84

TABLE 12.2 – *Résultats de recherche obtenus par NLGEN.Sp sur chaque élément de la totalité des IC décomposées en SVO.*

score de rappel pour toutes les langues. Ce même analyseur amélioré aiderait à supprimer le risque élevé de génération de phrases au contenu sémantique erroné indiqué par la précision (lorsque la *PS* est mal analysée et inadaptée sémantiquement, les algorithmes *NLGEN.replace* et *NLGEN.surface* sont incapables de l'exploiter) auquel conduit *NLGEN.S*. Néanmoins il est intéressant d'observer que dans cette expérience, jusqu'à de 60% des IC complexes pouvaient être correctement exprimées avec la PS retournée.

Comme on pouvait s'y attendre, l'algorithme *NLGEN.SP* qui n'est soumis à aucune contrainte de nature sémantique (la structure sémantique d'une structure *SVO* est implicite, et aucun analyseur n'est utilisé) rencontre beaucoup moins de problèmes de précision que *NLGEN.S*. Le structures *SVO* sont simples et suffisamment nombreuses dans le corpus d'apprentissage utilisé pour répondre aux besoins d'expression des IC. Les 8 à 10% d'erreurs observées sont imputables aux défauts introduits par les étiqueteurs lors de la phase d'apprentissage du CPM et sont acceptables pour les besoins d'un système prototype.

12.2.2 Évaluation de la génération

Pour évaluer les capacités des composants de génération (*NLGEN.replace* et *NLGEN.surface* dans notre architecture) et pour le cas de la génération par agrégation, celles obtenues lors l'assemblage préalable de structures *SVO* au sein d'une phrase correspondant à l'IC (*NLGEN.Agg*), nous ne conservons que les *phrases candidates (PC)* valides. Générer des phrases d'après des éléments phrastiques modèles erronées (PC et PC_{SVO}) n'apporterait en effet aucune information sur les performances opérationnelles de ces composants.

Évaluation	Syntaxe (%)			Sémantique (%)		
	Note 0 Erronée	Note 1 Erreurs	Note 2 Correcte	Note 0 Erroné	Note 1 Ambigu	Note 2 Conforme
Français	0	2	98	8	14	78
Anglais	0	3	97	6	17	77
Espagnol	5	10	85	10	30	60

TABLE 12.3 – *Résultats de génération obtenus par NLGEN.replace et NLGEN.surface pour des phrases complètes extraites par NLGEN.S.*

Les résultats de génération après réécriture de phrases complètes par un processus de remplacement et de traitement de surface sont indiqués dans le tableau 12.3. La quantité de défauts syntaxiques subsistants après une génération sont très faibles, et majoritairement imputables à un défaut d'application du modèle de langage lors de l'activation du composant *NLGEN.surface*. Les erreurs sémantiques plus nombreuses sont généralement dues à un défaut de remplacement des entités nommées co-référentes, consécutivement à un étiquetage erroné du corpus d'apprentissage.

Évaluation	Syntaxe (%)			Sémantique (%)		
	Note 0 Erronée	Note 1 Erreurs	Note 2 Correcte	Note 0 Erroné	Note 1 Ambigu	Note 2 Conforme
Français	2	8	90	5	14	81
Anglais	1	6	93	11	10	79
Espagnol	4	16	80	11	24	65

TABLE 12.4 – *Résultats de génération obtenus avec agrégation de SVO par NLGEN.agg suivi de NLGEN.replace et NLGEN.surface.*

12.3 Conclusion

Les résultats de génération par agrégation de proto-phrases décrits dans le tableau 12.4 laissent apparaître une introduction d'erreurs syntaxiques plus importante que pour la génération d'après une phrase complète. Ce phénomène est imputable à des tentatives d'agrégation de structures *SVO* incompatibles entre elles. Les erreurs de la famille des ambiguïtés sémantiques sont elles aussi imputables à des agrégations de proto-phrases dont certains éléments ne sont pas compatibles entre eux. Ces résultats nous paraissent décrire un bon niveau de performance qui permet de générer 80% de phrases correctes si l'on considère l'algorithme à base d'agrégation par *SVO* et 60% de phrases correctes avec un réemploi de *phrases support (PS)* complètes. L'algorithme de génération par phrase complète est néanmoins plus délicat à déployer - en l'absence d'un analyseur sémantique performant - puisqu'il succède à un processus d'extraction d'information qui ne retourne que 60% de propositions sémantiquement fiables. En l'état, on doit donc considérer que sans possibilité de rejeter les *phrases modèles (PM)* dont la représentation logique est inadaptée avant de lancer le processus de génération, l'algorithme *NLGEN.S* - bien que fonctionnel - serait difficile à exploiter dans une application de GAT opérationnelle. Il ouvre néanmoins des perspectives de recherches, très liées aux évolutions des méthodes d'analyse sémantique. En revanche, le générateur *SVO*, nous semble immédiatement exploitable. Nous l'avons implémenté sous une forme prototype dans le générateur SimpleNLG [5].

5. Voir Annexe B.

Conclusions et perspectives

Nous avons présenté dans ce mémoire un système de Génération Automatique de Texte (GAT) fonctionnant avec des méthodes statistiques. Il utilise pour générer des phrases un *Générateur de Surface* dont la particularité est d'utiliser des phrases préexistantes, contenues dans un *Corpus de Phrases Modèles (CPM)*. Le CPM est appris entièrement par une méthode automatique. Ces travaux reposent sur deux postulats. Qu'il soit en premier lieu possible de générer un texte en réutilisant des *phrases modèles (PM)* issues d'un corpus. Ensuite, que la modélisation fine des phrases d'un corpus par un étiquetage et une analyse sémantique rende la création automatique de CPM possible.

Contributions aux méthodes de GAT

Le système de GAT à base d'apprentissage automatique que nous avons mis au point fonctionne en langue anglaise, française et espagnole. Il remplace dans une architecture de GAT *pipeline* classique les grammaires du *composant de génération de surface* par des modèles de phrases - les PM - appris automatiquement. Les PM sont retrouvées dans le CPM en appliquant une méthode de *recherche d'information (RI)*. Notre proposition repose sur deux algorithmes principaux.

Le premier algorithme, $NLGEN.S$, tente de retrouver une phrase complètement formée qui puisse exprimer une *Intention de Communication (IC)*, quelle que soit la longueur ou la complexité de cette IC. Cet algorithme est intégré dans un composant de génération dont le rôle est de réutiliser et de transformer une phrase existante. Cette phrase existante, la *phrases support (PS)*, a reçu un étiquetage de *Part Of Speech*, entités nommées, co-références et logique sémantique interne formalisée avec la *Discourse Representation Theory (DRT)*. Le composant de génération qui implémente $NLGEN.S$ retrouve dans un CPM une PS compatible avec une IC. L'étiquetage fin est ensuite utilisé pour transformer la PS en remplaçant certains de ses éléments par ceux de l'IC. Une phase de posttraitement à base de modèles de langage permet de supprimer les erreurs mineures éventuellement introduites dans la surface de la phrase lors du processus d'insertion des éléments de IC.

Le second algorithme, $NLGEN.Sp$, dit de *Génération par agrégation de structures Sujet, Verbe, Objet* répond au problème de la rareté des PM disponibles pour exprimer les

IC complexes. Il permet également de rendre triviale la phase d'analyse sémantique et ainsi de supprimer les erreurs que cette dernière introduit parfois dans le processus de génération. Ce second algorithme consiste en un processus de GAT statistique robuste fonctionnant par agrégation de proto-phrases *Sujet, Verbe, Objet* (*SVO*). La principale différence de cet algorithme avec *NLGEN.S* est qu'il applique exclusivement le processus de recherche des PS et de formalisation des IC à des structures *SVO* qu'il agrège par la suite en une phrase complète. L'agrégation est réalisée avec trois règles d'agrégation par ellipses rhétoriques. La *Génération par agrégation de structures SVO* permet de répondre au manque de précision des méthodes actuelles d'analyse sémantique en diminuant la complexité du problème à résoudre.

Nous avons montré qu'il était possible d'exprimer une IC complexe avec une phrase existante mais que la rareté de ces phrases et le manque de précision des analyseurs sémantiques ne permettaient pas de répondre à tous les cas de figure. Nous avons complété notre proposition et amélioré sa robustesse en montrant qu'avec un nombre limité de règles d'agrégation et un corpus de structures *SVO* suffisamment vaste, il était permis d'envisager de produire des systèmes de GAT automatiquement. Même si on ne peut ignorer la pauvreté des structures stylistiques issues de phrases générées par trois règles d'agrégation par ellipses, il n'en demeure pas moins que ces phrases au contenu sémantique riche sont syntaxiquement correctes, linguistiquement acceptables et capables de véhiculer une information complexe et structurée. Ce système expérimental présente un niveau de performances qui souligne le potentiel des méthodes proposées.

Contributions aux méthodes d'étiquetages

Pour supporter le générateur à base de CPM, nous avons élaboré une architecture équipée de ressources lexicales et servie par un système d'étiquetage et d'analyse de corpus. Si certaines des technologies mises en œuvre peuvent être considérées comme arrivées à maturité, tel l'étiquetage morphosyntaxique, il n'en vas pas de même pour d'autres telles que l'analyse sémantique ou la détection de co-références. D'autres technologies considérées comme performantes par la communauté, tel l'étiquetage d'entités nommées, se sont révélées difficilement adaptables et impossibles à entraîner dans un contexte multilingue faute de ressources. Une partie importante de notre travail a donc consisté à élaborer des outils d'apprentissage et des ressources unifiées qui nous ont permis de développer un corpus multilingue finement étiqueté, exploitable par nos algorithmes de GAT.

Apports en corpus riches

Lorsque nous avons débuté ce travail en septembre 2007, les corpus de grande taille étiquetés et analysés, disponibles en plusieurs langues dans un champ sémantique suffisamment ouvert pour autoriser la conduite d'expériences de génération multilingue n'étaient pas encore largement diffusés. L'encyclopédie Wikipédia, qui n'existe sous sa forme actuelle que depuis 2002, et avec un contenu multilingue suffisant depuis moins

de 5 ans[6], a été l'un des premiers éléments clé de cette démarche. En 2010, le corpus textuel de Wikipédia est une ressource acceptée et appréciée par la communauté du TAL pour de nombreuses applications. Elle fait l'objet de publications et d'expérimentations régulières.

Durant la même période, de nouveaux corpus finement étiquetés, élaborés d'après une intégration de ressources multiples (Penn TreeBank, Wall Street Journal, Brown Corpus), ont été mis au point dans le cadre de la campagne d'évaluation CoNLL 2008 (Surdeanu et al., 2008). D'abord disponible uniquement en anglais, le corpus de CoNLL 2008 a été complété lors de la campagne 2009 (Hajič et al., 2009), de versions catalane, chinoise, tchèque, allemande, japonaise et espagnole. Ces corpus - qui correspondent en tous points à nos ressources quant aux informations qu'ils contiennent (POS, EN, analyse sémantique, etc.) - sont encore insuffisants pour répondre aux besoins de notre architecture (4393 phrases en version japonaise, 40 000 phrases en anglais, allemand, tchèque et 15 à 20 000 phrases en catalan, chinois, espagnol).

La ressource NLGbAse avec 30 millions de phrases étiquetées en anglais, 10 millions en français, 4 millions en espagnol demeure donc toujours novatrice. Intrinsèquement, elle ne diffère que par la qualité de son étiquetage du corpus CoNLL - étiqueté manuellement - et peut donc encore bénéficier d'importantes améliorations.

Méthodes d'étiquetage

Nous avons élaboré un étiqueteur statistique d'entités nommées robuste et facile à entraîner dans plusieurs langues. Nous avons proposé une méthode de génération automatique de corpus d'entraînement pour cet étiqueteur en utilisant des contenus encyclopédiques et des *métadonnées*. L'étiqueteur d'entités nommées et les *métadonnées* ont servi de base pour développer un système original de détection de co-références. Ces deux systèmes ont été évalués dans des campagnes scientifiques et ont obtenu de bons résultats.

Limites du système proposé

La méthode décrite ne prend pas en compte pour le moment certains aspects linguistiques tels que la négation, les modes (conditionnel, impératif) et les propositions multiples. Il conviendrait de les introduire au niveau de l'étiquetage du CPM dans une évolution ultérieure. Globalement, les performances du système présenté sont hautement dépendantes du fonctionnement des composants d'étiquetage et d'analyse déployés lors de la phase de construction du CPM. Nous avons observé que les erreurs introduites dans les phrases générées par le système étaient dues à des défectuosités mineures des composants d'analyse et d'étiquetage influençant la phase de modélisation des PM. Comme tout système à base d'apprentissage automatique, notre proposition possède un potentiel de progression. Nous estimons que ce potentiel de progres-

6. http://en.wikipedia.org/wiki/History_of_Wikipedia

sion est particulièrement lié aux évolutions futures des méthodes d'analyse sémantique et de détection de co-références.

Nous avons observé, au cours des trois dernières années, une constante amélioration des performances de nos algorithmes de génération. Ces progrès réguliers découlaient directement de l'amélioration qualitative des ressources produites (*métadonnées*, corpus étiquetés par EN) par de nouveaux outils de TAL (en particulier les analyseurs sémantiques diffusés consécutivement aux *Shared Tasks* CoNLL 2008 et 2009).

Atouts du système proposé

Le système dans sa version expérimentale produit parfois des phrases légèrement approximatives mais il restitue correctement le sens initial de l'IC. On peut donc considérer qu'il est utilisable dans un contexte appliqué s'il est correctement entraîné avec un corpus offrant des phrases représentatives d'un domaine sémantique précis. Il offre alors une alternative viable aux solutions traditionnelles de GAT à base de grammaires mises au point manuellement. Il peut donner une réponse peut coûteuse à une problématique de génération multilingue.

Perspectives

Dans la continuité directe des travaux présentés dans cette thèse, plusieurs évolutions semblent s'imposer et seront réalisées dans un futur proche.

En premier lieu, l'analyseur sémantique étant stratégique dans l'architecture proposée, l'amélioration de la fiabilité de ce composant augmente les performances des algorithmes de génération. Il sera donc essentiel de suivre les progrès qui seront obtenus dans ce domaine et d'intégrer les dernières innovations dans le *pipeline* d'apprentissage automatique pour en mesurer l'influence sur la génération.

La question de l'utilisation d'une structure intermédiaire entre la phrase complexe et la proto-phrase *SVO* méritera aussi d'être posée. Nous souhaitons en particulier savoir jusqu'à quel point un analyseur syntaxique en dépendances associé à un analyseur sémantique performant nous permettrait de décomposer des phrases en séquences intermédiaires plus complexes que le *SVO*. Il serait intéressant d'évaluer dans quelle mesure ces structures phrastiques intermédiaires seraient réutilisables dans une architecture adaptée. Ce travail sur la structure des phrases pourrait avoir d'autres implications. Nous voulons en particulier mener des expériences pour tenter de récupérer automatiquement depuis un corpus, non plus des modèles de phrases mais une connaissance grammaticale (les morphologies, les conjugaisons de verbes). Nous voudrions intégrer cette connaissance automatiquement dans une architecture de GAT à base de règles telle que celle de SimpleNLG. Il y aurait là une voie intermédiaire entre notre architecture entièrement statistique et celle des systèmes classiques.

L'évaluation précise des capacités de génération dans des domaines sémantiques fermés nous paraît indispensable. Dans ce travail, nous avons mené des expériences sur un domaine relativement ouvert, plus difficile que dans un cadre sémantique restreint mais aussi plus révélateur des limites théoriques de notre système. Dans une perspective applicative, il serait intéressant d'élaborer des CPM dédiés et de les évaluer. Le prototype de librairie de génération en langage Java que nous sommes en train d'achever sera l'un des instruments de cette démarche expérimentale complémentaire.

Nous envisageons aussi d'étudier la possibilité d'intégrer un module de génération dans une architecture de Résumé Automatique (RA). En particulier la possibilité d'utiliser nos algorithmes pour la re-génération et l'agrégation de phrases après leur extraction dans le contexte des propositions de (Cunha et al., 2009). Nous aimerions aussi vérifier la capacité du générateur à produire des phrases comparables à celles d'un algorithme de compression d'un système de RA (Loupy et al., 2010).

Implications théoriques

Ces travaux nous amènent à évoquer la question de la relation entre une *Intention de Communication* et une suite de séquences *Sujet, Verbe, Objet* assemblées par des règles d'agrégation. Il a été proposé que tous les langages respectaient cet ordre en tant que structure de base (Kayne, 1994) et l'importance de ce phénomène à été admise et prise en compte par Chomsky dans la dernière évolution de son corpus théorique [7]. Les plus récents développements en psycholinguistiques (Langus et Nespo, 2010) envisagent désormais de relier cette structure à la constitution biologique de l'organe auditif et à sa sensibilité à la prosodie. On pourrait par exemple étudier le degré de sophistication des règles impliquées, et leur dépendance avec le niveau de culture ou de maturité possédés par le sujet qui les déploie. Ce questionnement pourrait peut être trouver des réponses dans l'abondant corpus scientifique traitant de la formation des Pidgins [8], ces proto-langages qui apparaissent spontanément dès que deux communautés sans langue commune entrent en contact.

Nous avons montré qu'avec seulement trois règles élémentaires d'assemblage, nous pouvions dès à présent assembler dans 80% des cas des phrases proches de celles produites par un système de GAT à base de règles, en nous dispensant complètement de la phase de création manuelle de ce système. Ce processus ressemble beaucoup à celui qui préside à l'acquisition du langage chez un enfant qui, sans aucune connaissance préalable des règles de grammaire et de la syntaxe, est en mesure de produire au fil de son apprentissage, des structures de plus en plus élaborées et complexes en partant des plus simples. (Clark et Chouinard, 2002) ont montré expérimentalement comment des interactions entre un enfant et un adulte, faites de corrections et de reformulations successives, mènent progressivement à une acquisition des mécanismes de production de discours [9]. Plusieurs études récemment menées par des primatologues

7. (Chomsky, 1995) pages 340, 335.
8. Lire par exemple *Génèse des Pidgins et des Créoles* de Maurice Houis.
9. Un rapport interne plus détaillé sur les travaux de Clark et Chouinard est fourni à l'adresse http://www.stanford.edu/dept/linguistics/colloq/prev/2002/papers/clark-all.

tendent à démontrer que le concept de syntaxe, vu comme l'assemblage de séquences phonétiques pour obtenir une structure plus complexe sémantiquement que les éléments phonétiques pris isolément, est présent dans le règne animal. A plusieurs degrés de l'évolution (Yip, 2006), chez les singes Mona (Riede et al., 2006) ou maintenant de manière très précise chez les singes Campbell (Ouattara et al., 2009), il a été observé des séquences vocales assemblées, décrites comme possibles proto-phrases. Ces observations conduisent de nombreux linguistes et psycho-linguistes à émettre l'hypothèse que le langage humain pourrait n'être finalement qu'une forme complexifiée d'une fonction biologique naturelle, présente à tous les stades de l'évolution (Lieberman et al., 1972; Pinker et Jackendoff, 2005; Jackendoff et Pinker, 2005; Evans et Levinson, 2009). (Hauser et al., 2002) citant plusieurs recherches, en viennent à admettre que les *études qui utilisent des méthodes d'apprentissages classiques comme celles utilisant les compétences spontanées révèlent que les animaux acquièrent et utilisent de nombreux concepts abstraits* [10]. Ils envisagent même de réduire la spécificité du langage chez les hommes à la seule récursivité [11].

Cette extension au champ biologique de la fonction langagière pourrait avoir une implication dans nos recherches. En particulier si la faculté de langage venait à être admise comme présente à plusieurs stades évolutifs dans l'arbre du vivant, sous une forme rudimentaire d'abord (le lexique-signal), ensuite combinée (la proto-syntaxe puis la proto-phrase) et pour finir assemblée de manière toujours plus complexe (la phrase composée d'unités de sens agrégées, au besoin récursivement) par des règles informatiquement reproductibles. Prenant acte de ces nouvelles voies que la théorie linguistique pourrait explorer, nous pensons que les systèmes de Génération Automatique de Texte peuvent occuper une place de choix dans l'étude du langage. Nous émettons aussi l'hypothèse que si les mécanismes de production langagière sont issus d'un lent processus évolutif et biologique, dont on découvre aujourd'hui des traces dans la nature à des degrés divers, il est peut-être possible d'en modéliser certains aspects humains par observation automatisée de corpus. Nous souhaitons en particulier vérifier jusqu'à quel point toute structure syntaxique de surface exprimant une intention de communication complexe, peut être reproduite par des règles d'assemblage et d'agrégation appliquées à structures simples, règles qui seraient alors à inventorier.

pdf et il est également possible de consulter le corpus CHILDES, issu de ces recherches, sur http://childes.psy.cmu.edu/data/.

10. *«Studies [...] reveal that animals acquire and use a wide range of abstract concepts, including tool, color, geometric relationships, food, and number».*

11. (Hauser et al., 2002) page 1569 *«it is important to distinguish between questions concerning language as a communicative system and questions concerning the computations underlying this system, such as those underlying recursion».*

Annexe A

Publications de l'auteur reliées à cette recherche

A.1 Revue internationale avec comité de lecture

Canadian Journal of Information and Library Science (RCSIB) - (article accepté Juillet 2009. à paraître) - Modélisation automatique de connecteurs logiques par analyse statistique du contexte - *Eric Charton, Juan Manuel Torres-Moreno LIA (Laboratoire Informatique d'Avignon)* - in Numéro thématique Fouille de texte et Recherche d'information.

A.2 Communications de conférences internationales avec comités de lecture

MOG 2010 (ACL-SIGGEN) - Juillet 2010 - A Preprocessing System to Include Imaginative Animations According to Text in Educational Applications *Eric Charton, Michel Gagnon, Benoit Ozell (École Polytechnique de Montréal) - Dublin, Irland, 6 July 2010.*

ICASSP-10 (IEEE) - Mars 2010 - Unsupervised knowledge acquisition for extracting Named Entities from Speech. *Frédéric Béchet (Laboratoire d'Informatique Fondamentale de Marseille), Eric Charton LIA (Université d'Avignon et des Pays de Vaucluse) - Dallas, Texas.*

LREC 2010 - Mai 2010 - NLGbAse : a free linguistic resource for Natural Language Processing systems. *Eric Charton and Juan Manuel Torres-Moreno, LIA (Laboratoire Informatique d'Avignon) - Valetta, Malta.*

ECIR-08 - Mars 2008 - Annotation of Scientific Summaries for Information Retrieval. *Ibekwe-SanJuan F., Fernandez S., SanJuan E., Charton E, Glasgow, Scotland.*

A.3 Communications de conférences nationales avec comités de lecture

TALN 2010 - Juillet 2010 - Extension d'un système d'étiquetage d'entités nommées en étiqueteur sémantique. *Eric Charton, Michel Gagnon, Benoit Ozell, GIGL (École Polytechnique de Montréal)[oral] - Montréal.*

TALN 2009 - Juin 2009 - Classification d'un contenu encyclopédique en vue d'un étiquetage par entités nommées. *Eric Charton et Juan Manuel Torres-Moreno - LIA (Université d'Avignon et des Pays de Vaucluse)[oral] - Senlis.*

RECITAL-09 - Juin 2009 - Combinaison de contenus encyclopédiques multilingues pour une reconnaissance d'entités nommées en contexte. *Eric Charton - LIA (Université d'Avignon et des Pays de Vaucluse)[poster] - Senlis.*

JADT-08 - Mars 2008 - Réécriture automatique de phrases par modèle de langage. *Charton Eric, Torres-Moreno Juan-Manuel et San-Juan Eric, LIA (Université d'Avignon et des Pays de Vaucluse) - Lyon.*

A.4 Campagnes d'évaluation scientifiques

INLG 2010 (ACL-SIGGEN) (GREC challenge) - Juillet 2010 - Poly-co : an unsupervised coreference detection system. *Eric Charton, Michel Gagnon and Benoit Ozell, GIGL (École Polytechnique de Montréal) - Dublin, Irland.*

ESTER 2 - Avril 2009 - Participation du Lia à la campagne d'évaluation ESTER 2, Entités nommées. *Frédéric Béchet et Eric Charton, LIA (Université d'Avignon et des Pays de Vaucluse).*

TALN08 - Deft08 - Juin 2008 - Pré-traitements classiques ou par analyse distributionnelle : application aux méthodes de classification automatique déployées pour DEFT08. *Eric Charton, Nathalie Camelin, et al. LIA (Université d'Avignon et des Pays de Vaucluse).*

A.5 Séminaires et conférences invités

Workshop African HLT - Janvier 2010 - Djibouti - Application de la Génération Automatique de textes au traitement des langues Africaines, *Eric Charton. GIGL (École Polytechnique de Montréal).*

Séminaire Rali - Février 2010 - Université de Montréal / Montréal - Proposition d'architecture à base de corpus pour la Génération Automatique de Texte, *Eric Charton. GIGL (École Polytechnique de Montréal).*

Séminaire LIA - Mai 2008 - Université d'Avignon / Avignon - État de l'art de la génération automatique de textes, *Eric Charton. LIA (Université d'Avignon et des Pays de Vaucluse).*

Annexe B

Système prototype

B.1 NLGbAse

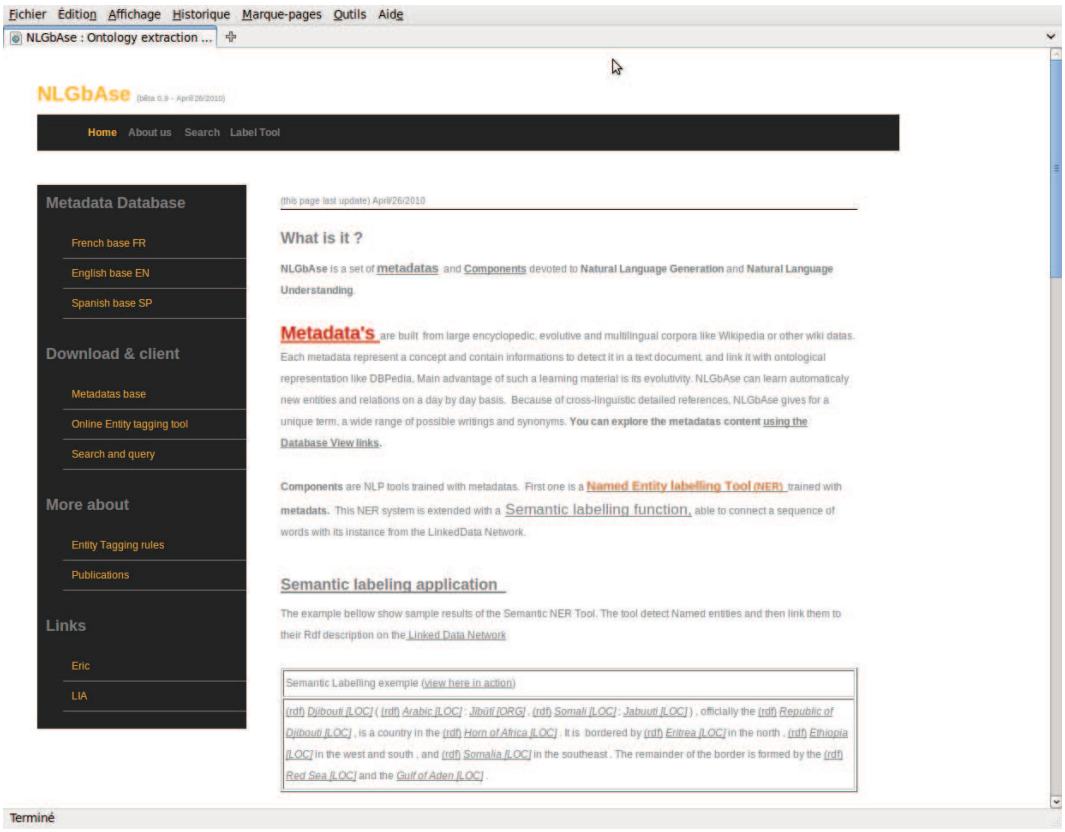

FIGURE B.1 – *Interface en ligne du module d'apprentissage et de ressources NLGbAse accessible par* `http://www.nlgbase.org`.

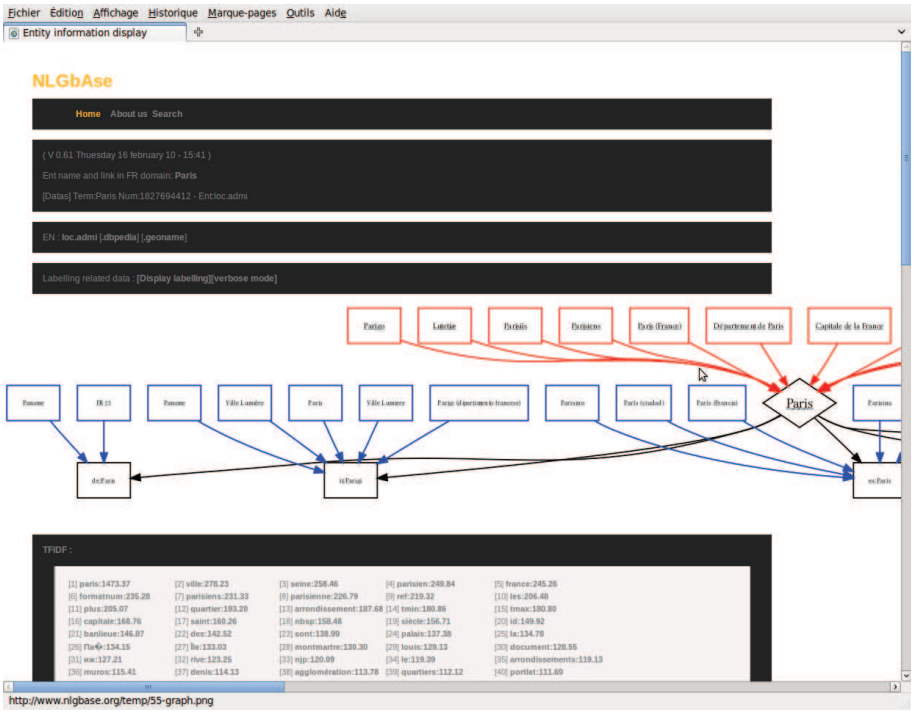

FIGURE *B.2* – *Exemple de métadonnée contenue dans NLGbAse.*

FIGURE *B.3* – *Démonstration de la chaîne d'étiquetage de NLGbAse.*

B.2 NLGEN

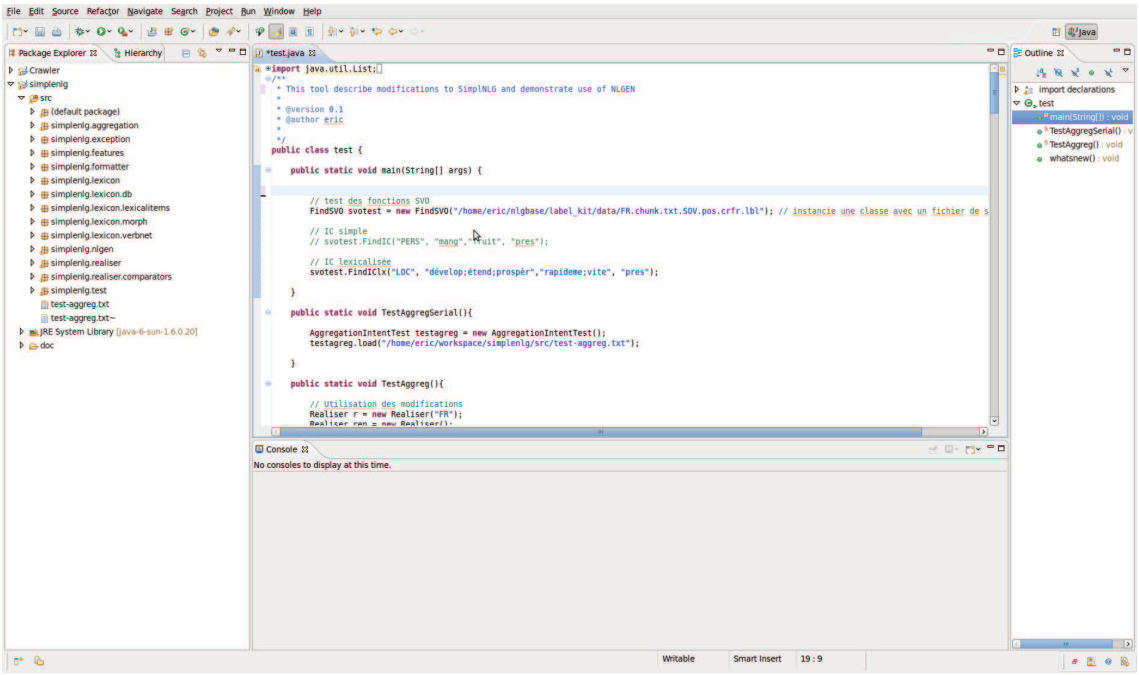

FIGURE *B.4* – *Le générateur NLGEN est implémenté sous forme de classes complémentaires du générateur SimpleNLG.*

B.2.1 Exemples d'utilisation de NLGEN

La classe d'aggrégation **AggregationGeneric**.

```
// ——————————————————————————————
// Instanciation des classes
// ——————————————————————————————
Realiser r = new Realiser("FR");
AggregationGeneric aggregator = new AggregationGeneric();
SPhraseSpec result = new SPhraseSpec();

// ——————————————————————————————
// Test d'aggrégation f
// ——————————————————————————————

SPhraseSpec is1 = new SPhraseSpec("depuis deux ans ma soeur", "vit", "à paris");
SPhraseSpec is2 = new SPhraseSpec("depuis deux ans ma soeur", "travaille", "à la défense");
// NOTENSE permet de désactiver les règles anglaises de SimpleNLG
is1.setTense(Tense.NOTENSE);
is2.setTense(Tense.NOTENSE);

result = aggregator.apply(is1, is2);

String output1 = r.realiseDocument(result);
System.out.println(output1); //

SPhraseSpec ds1 = new SPhraseSpec("Henri", "lit", "un livre");
SPhraseSpec ds2 = new SPhraseSpec("Henri", "regarde", "la tv");
SPhraseSpec ds3 = new SPhraseSpec("Henri", "réfléchit", "à haute voix");

ds1.setTense(Tense.NOTENSE);
ds2.setTense(Tense.NOTENSE);
ds3.setTense(Tense.NOTENSE);

result = aggregator.apply(ds1, ds2, ds3);

String output2 = r.realiseDocument(result);
System.out.println(output2); // .
```

TABLE *B.1* – *Définition d'un agrégateur.*

Structure	Phrase assemblée
is1+is2	`Depuis deux ans ma soeur vit à paris et travaille à la défense.`
ds1+ds2+ds3	`Henri lit un livre, regarde la tv, et réfléchit à haute voix.`

TABLE *B.2* – *Résultats d'agrégation produit par B.1.*

La classe de recherche de structures SVO **FindSVO**.

```
public static void main(String[] args)
{

// ———————————————————————————
// Instanciation des classes
// ———————————————————————————
Realiser r = new Realiser("FR");
String output;
AggregationGeneric aggregator = new AggregationGeneric();
SPhraseSpec result1 = new SPhraseSpec();
SPhraseSpec result2 = new SPhraseSpec();


// test des fonctions SVO
// instancie une classe associée à un CPM
FindSVO svotest = new FindSVO("FR.chunk.txt.SOV.pos.crfr.lbl");
svotest.setTense(Tense.PRESENT);

// IC simple
result1 = svotest.FindIC("PERS", "mang","fruit");
Surfacer s1 = new Surfacer(result1,"PERS=Jean");
result1 = s1.transformed();
output = r.realiseDocument(result1);
System.out.println(output); //


// IC lexicalisée
result2 = svotest.FindIClx("LOC", "prospèr","vite");
// La lexicalisation ajoute ("LOC", "étend ;prospèr ;dévelop ;","rapide ;vite");
Surfacer s2 = new Surfacer(result2,"LOC=Lyon");
result2 = s2.transformed();
output = r.realiseDocument(result2);
System.out.println(output); //

}
```

TABLE *B.3* – *Définition d'un générateur pour deux phrases utilisant le CPM_{svo}.*

Phrase modèle localisée	Phrase générée
result1=*Thalès mange alors un fruit*	`Jean mange alors un fruit.`
result2=*L'abbaye d'Ardenne se développe rapidement*	`Lyon se développe rapidement.`

TABLE *B.4* – *Génération de result1 et result2 selon B.3.*

Annexe C

Licence de diffusion

Paternité - Pas d'Utilisation Commerciale - Partage à l'Identique 2.0 France (CC BY-NC-ND 2.0)

Vous êtes libres de :

partager — reproduire, distribuer et communiquer l'oeuvre

Selon les conditions suivantes :

Paternité — Vous devez attribuer l'oeuvre de la manière indiquée par l'auteur de l'oeuvre ou le titulaire des droits (mais pas d'une manière qui suggérerait qu'ils vous soutiennent ou approuvent votre utilisation de l'oeuvre).

Pas d'utilisation commerciale — Vous n'avez pas le droit d'utiliser cette oeuvre à des fins commerciales.

Pas de travaux dérivés — Vous n'avez pas le droit de modifier, de transformer ou d'adapter cette œuvre.

http://creativecommons.org/licenses/by-nc-nd/2.0/fr/

Liste des illustrations

Liste des tableaux

Bibliographie

(Adorni et Zocks, 1993) G. Adorni et M. Zocks, 1993. Trends in natural language generation : an artificial intelligence ... Dans les actes de *EWNLG'93*.

(Anis, 1994) J. Anis, 1994. Ordinateurs et traduction : survol d'un demi-siècle. *Langages (Paris) 28*(116), 111–122.

(Arnold et al., 1993) D. Arnold, T. Badia, J. van Genabith, S. Markantonatou, S. Momma, L. Sadler, et P. Schmidt, 1993. Experiments in Reusability of Grammatical Resources. *Proceedings of the sixth conference on European chapter of the Association for Computational Linguistics -*, 12.

(Aubert, 2002) X. Aubert, 2002. An overview of decoding techniques for large vocabulary continuous speech recognition. *Computer Speech & Language 16*(1), 89–114.

(Auer et al., 2007) S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, et Z. Ives, 2007. DBpedia : A Nucleus for a Web of Open Data. Dans les actes de *In 6th Int'l Semantic Web Conference, Busan, Korea*, 11–15. Springer.

(Bangalore et Rambow, 2000a) S. Bangalore et O. Rambow, 2000a. Corpus-based lexical choice in natural language generation. Dans les actes de *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, 471. Association for Computational Linguistics.

(Bangalore et Rambow, 2000b) S. Bangalore et O. Rambow, 2000b. Exploiting a probabilistic hierarchical model for generation. Dans les actes de *Proceedings of the 18th conference on Computational linguistics*, 42–48.

(Bateman, 1990) J. A. Bateman, 1990. Upper modeling : Organizing knowledge for natural language processing. Dans les actes de *Association for Computational Linguistics*, 54–61. ACL.

(Bateman, 1997) J. A. Bateman, 1997. Enabling technology for multilingual natural language generation : the KPML development environment. *Natural Language Engineering 3*.

(Béchet et al., 2000) F. Béchet, A. Nasr, et F. Genet, 2000. Tagging unknown proper names using decision trees. Dans les actes de *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, 84. Association for Computational Linguistics.

(Belz, 2005) A. Belz, 2005. Statistical generation : Three methods compared and evaluated. Dans les actes de *Proceedings of the 10th European Workshop on Natural Language Generation (ENLG'05)*, Volume 05pages, 15–23.

(Belz, 2006) A. Belz, 2006. High-quality Probabilistic Generation of Language. *itri.brighton.ac.uk*.

(Belz et Kow, 2010) A. Belz et E. Kow, 2010. The GREC Challenges 2010 : Overview and Evaluation Results. *INLG 2010 Dublin*.

(Belz et al., 2009a) A. Belz, E. Kow, et J. Viethen, 2009a. The GREC named entity generation challenge 2009 : overview and evaluation results. Dans les actes de *Proceedings of the 2009 Workshop on Language Generation and Summarisation*, Numéro August, 88–98. Association for Computational Linguistics.

(Belz et al., 2009b) A. Belz, E. Kow, J. Viethen, et A. Gatt, 2009b. The GREC main subject reference generation challenge 2009 : overview and evaluation results. Dans les actes de *Proceedings of the 2009 Workshop on Language Generation and Summarisation*, Numéro June, 79–87. Association for Computational Linguistics.

(Benamara, 2004) F. Benamara, 2004. Generating Intensional Answers in Intelligent Question Answering Systems.

(Bikel et al., 1999) D. Bikel, R. Schwartz, et R. Weischedel, 1999. An algorithm that learns whats in a name. *Machine learning 7*.

(Borthwick et al., 1998) A. Borthwick, J. Sterling, E. Agichtein, et R, 1998. Exploiting diverse knowledge sources via maximum entropy in named entity. *Proc. of the Sixth*, 152–160.

(Bos, 2008) J. Bos, 2008. Let's not argue about semantics. *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08), Marrakech, Morocco*, 2835–2840.

(Brown et al., 1990) P. Brown, J. Cocke, S. Pietra, V. Pietra, F. Jelinek, J. Lafferty, R. Mercer, et P. Roossin, 1990. A statistical approach to machine translation. *Computational linguistics 16*(2), 85.

(Buchholz et Marsi, 2006) S. Buchholz et E. Marsi, 2006. CoNLL-X shared task on multilingual dependency parsing. *International Conference On Computational Linguistics*, 149–164.

(Bunescu et Pasca, 2006) R. Bunescu et M. Pasca, 2006. Using encyclopedic knowledge for named entity disambiguation. Dans les actes de *Proceedings of EACL*, Volume 6.

(Busemann et Horacek, 1998) S. Busemann et H. Horacek, 1998. A flexible shallow approach to text generation. Dans les actes de *Proceedings of the Ninth International Workshop on Natural Language Generation*, Numéro 2945, 238–247.

(Callison-Burch et al., 2006) C. Callison-Burch, M. Osborne, et P. Koehn, 2006. Re-evaluating the role of BLEU in machine translation research. *Proceedings of EACL*.

(Charton et al., 2008) E. Charton, N. Camelin, R. Acuna-agost, P. Gotab, R. Lavalley, R. Kessler, et S. Fernandez, 2008. Pré-traitements classiques ou par analyse distributionnelle :application aux méthodes de classification automatique déployées pour DEFT08. Dans les actes de *Deft 2008*.

(Charton et al., 2010a) E. Charton, M. Gagnon, et B. Ozell, 2010a. A preprocessing system to include imaginative animations according to text in educational applications. Dans les actes de *Mog 2010*, Dublin.

(Charton et al., 2010b) E. Charton, M. Gagnon, et B. Ozell, 2010b. Poly-co : an unsupervised co-reference detection system. Dans A. Belz et E. Kow (Eds.), *INLG 2010-GREC*, Dublin. ACL SIGGEN.

(Charton et Torres-Moreno, 2009) E. Charton et J. Torres-Moreno, 2009. Classification d'un contenu encyclopédique en vue d'un étiquetage par entités nommées. Dans les actes de *Taln 2009*, Volume 1, 24–26. TALN.

(Charton et Torres-Moreno, 2010) E. Charton et J. Torres-Moreno, 2010. Modélisation automatique de connecteurs logiques par analyse statistique du contexte. *Revue Canadienne de Sciences et de Bibliothéconomie (RCSIB) / Canadian Journal of Information and Library Science*.

(Charton et al., 2008) E. Charton, J. Torres-Moreno, et E. SanJuan, 2008. Réécriture statistique de phrases basée sur des modèles de langage. Dans les actes de *JADT 2008*, 1–11.

(Chen et al., 2002) J. Chen, S. Bangalore, O. Rambow, et M. a. Walker, 2002. Towards automatic generation of natural language generation systems. *Proceedings of the 19th international conference on Computational linguistics -*, 1–7.

(Chomsky, 1957) N. Chomsky, 1957. *Syntactic structures* (2de ed.). de Gruyter Mouton.

(Chomsky, 1969) N. Chomsky, 1969. *Aspects of the Theory of Syntax* (1st ed.). Numéro 1st Paperback Ed. edition (March 15, 1969). The MIT Press.

(Chomsky, 1986) N. Chomsky, 1986. *Knowledge of language : its nature, origin, and use*. Greenwood Publishing.

(Chomsky, 1995) N. Chomsky, 1995. *The minimalist program*. MIT Press.

(Chomsky, 2005) N. Chomsky, 2005. Three Factors in Language Design. *Linguistic inquiry 36*(1), 1–22.

(Clark et Chouinard, 2002) E. V. Clark et M. M. Chouinard, 2002. Énoncés enfantins et reformulations adultes dans l'acquisition du langage. *Langages 34*, 9–23.

(Corston-Oliver et al., 2002) S. Corston-Oliver, R. Moore, M. Gamon, et E. Ringger, 2002. An overview of Amalgam : A machine-learned generation module.

(Cunha et al., 2009) I. D. Cunha, J. Torres-Moreno, et P. Velazquez, 2009. Un algoritmo lingüístico-estadístico para resumen automático de textos especializados. *Linguamática 2*, 67–79.

(Dale et Viethen, 2009) R. Dale et J. Viethen, 2009. Referring expression generation through attribute-based heuristics. Dans les actes de *Proceedings of the 12th European Workshop on Natural Language Generation*, Numéro March, 58–65. Association for Computational Linguistics.

(Danlos et al., 2000) L. Danlos, G. Lapalme, et V. Lux, 2000. Generating a Controlled Language. Dans les actes de *Proceedings of the first international conference on Natural language generation-Volume 14*, 147. Association for Computational Linguistics.

(Danlos et Roussarie, 2000) L. Danlos et L. Roussarie, 2000. La Génération Automatique de Textes. *Ingéniérie de la langue, Hermès*.

(Deemter et al., 2005) K. V. Deemter, M. Theune, et E. Krahmer, 2005. Real versus Template-Based Natural Language Generation : A False Opposition ? *Computational Linguistics 31*(1), 15–24.

(Doddington et al., 2004) G. Doddington, A. Mitchell, M. Przybocki, L. Ramshaw, S. Strassel, et R. Weischedel, 2004. The automatic content extraction (ACE) program–tasks, data, and evaluation. Dans les actes de *Proceedings of LREC*, Volume 4, 837–840. Citeseer.

(Elhadad et Robin, 1996) M. Elhadad et J. Robin, 1996. An overview of SURGE : A reusable comprehensive syntactic realization component. Dans les actes de *Eighth International Natural Language Generation Workshop. Demonstrations and Posters*, 1–4. Citeseer.

(Etchegoyhen et Wehrle, 1998) T. Etchegoyhen et T. Wehrle, 1998. System Demonstration Overview Of GBGen.

(Evans et Levinson, 2009) N. Evans et S. C. Levinson, 2009. The myth of language universals : language diversity and its importance for cognitive science. *The Behavioral and brain sciences 32*(5), 429–48 ; discussion 448–494.

(Fitch et al., 2005) W. T. Fitch, M. D. Hauser, et N. Chomsky, 2005. The evolution of the language faculty : clarifications and implications. *Cognition 97*(2), 179–210 ; discussion 211–25.

(Friedman, 1969) J. Friedman, 1969. Directed random generation of sentences. *Communications of the ACM 12*(1).

(Gagnon et Lapalme, 1996a) M. Gagnon et G. Lapalme, 1996a. From conceptual time to linguistic time. *Computational linguistics* (514), 1–43.

(Gagnon et Lapalme, 1996b) M. Gagnon et G. Lapalme, 1996b. Pretexte : A Generator for the expression of temporal information. *Lecture Notes in Artificial Intelligence*, 238–259.

(Galliano et al., 2009) S. Galliano, G. Gravier, et L. Chaubard, 2009. The ESTER 2 Evaluation Campaign for the Rich Transcription of French Radio Broadcasts. Dans les actes de *International Speech Communication Association conference 2009*, 2583–2586. Interspeech 2010.

(Gatt et al., 2008) A. Gatt, A. Belz, et E. Kow, 2008. The TUNA challenge 2008 : Overview and evaluation results. Dans les actes de *Proceedings of the Fifth International Natural Language Generation Conference*, 198–206. Association for Computational Linguistics.

(Grouin et al., 2008) C. Grouin, J.-b. Berthelin, S. E. Ayari, M. Hurault-plantet, et S. Loiseau, 2008. Présentation de DEFT'08 (DÉfi Fouille de Textes). *Actes de DEFT 2008 08*, 9–13.

(Guo et al., 2008) Y. Guo, J. Van Genabith, et H. Wang, 2008. Dependency-based n-gram models for general purpose sentence realisation. Dans les actes de *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, Numéro 2000, 297–304. Association for Computational Linguistics.

(Habash, 2004) N. Habash, 2004. The use of a structural n-gram language model in generation-heavy hybrid machine translation. *Natural Language Generation*, 61–69.

(Hajič et al., 2009) J. Hajič, M. Ciaramita, et R. Johansson, 2009. The CoNLL-2009 shared task : syntactic and semantic dependencies in multiple languages. *International Conference On Computational Linguistics*, 1–18.

(Hankash, 2009) P. Hankash, 2009. *Génération automatique de textes par satisfaction de contraintes.* Thèse de Doctorat, Paris VII.

(Harbusch et Kempen, 2009) K. Harbusch et G. Kempen, 2009. Generating clausal coordinate ellipsis multilingually : A uniform approach based on postediting. Dans les actes de *Proceedings of the 12th European Workshop on Natural Language Generation*, Numéro March, Morristown, NJ, USA, 138–145. Association for Computational Linguistics.

(Haspelmath, 2000) M. Haspelmath, 2000. Coordination. *Language typology and linguistic description 43*(9), 488.

(Hauser et al., 2002) M. Hauser, N. Chomsky, et W. Fitch, 2002. The faculty of language : What is it, who has it, and how did it evolve ? *Science 298*(5598), 1569.

(Hobbs, 1978) J. R. Hobbs, 1978. Resolving Pronoun References. *Lingua 44*(4), 311–338.

(Hori et Nakamura, 2006) T. Hori et A. Nakamura, 2006. An extremely large vocabulary approach to named entity extraction approach from speech. *2006 IEEE International Conference on Acoustics Speed and Signal Processing Proceedings*, I–973–I–976.

(Hovy, 1988) E. H. Hovy, 1988. Generating Natural Language under Pragmatic Constraints.

(Jackendoff, 1992) R. Jackendoff, 1992. *Semantic Structures*.

(Jackendoff et Pinker, 2005) R. Jackendoff et S. Pinker, 2005. The nature of the language faculty and its implications for evolution of language (Reply to Fitch, Hauser, and Chomsky). *Cognition 97*(2), 211–225.

(Jan van Eijck, 2005) Jan van Eijck, 2005. Discourse representation theory. *Lecture Notes in Computer Science* (1), 84–111.

(Johansson et Nugues, 2008) R. Johansson et P. Nugues, 2008. Dependency-based syntactic-semantic analysis with PropBank and NomBank. *Proceedings of the Twelfth Conference on Computational Natural Language Learning - CoNLL '08* (August), 183.

(Joshi et Rambow, 2003) A. Joshi et O. Rambow, 2003. A formalism for dependency grammar based on tree adjoining grammar. *Proceedings of MTT 2003*, 16–18.

(Jurafsky et al., 2000) D. Jurafsky, J. Martin, A. Kehler, K. Vander Linden, et N. Ward, 2000. *Speech and language processing*. Prentice Hall New York.

(Kamp, 1988) H. Kamp, 1988. Discourse representation theory. *Lecture Note in Computer Science 320/1988*, 84–111.

(Kan et McKeown, 2002) M. Kan et K. McKeown, 2002. Corpus-trained text generation for summarization. Dans les actes de *Proceedings of INLG 2002*, 1–8. Citeseer.

(Kaplan et Bresnan, 1987) R. M. Kaplan et J. Bresnan, 1987. Lexical Functional Grammar : a formal System for Grammatical Representation. *Decision Support Systems 3*(3), 269.

(Katz, 1987) S. M. Katz, 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer.

(Kayne, 1994) R. S. Kayne, 1994. *The antisymmetry of syntax*. MIT Press.

(Kazama et Torisawa, 2007) J. Kazama et K. Torisawa, 2007. Exploiting Wikipedia as external knowledge for named entity recognition. Dans les actes de *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 698–707.

(Knight et Hatzivassiloglou, 1995) K. Knight et V. Hatzivassiloglou, 1995. Two-level, many-paths generation. *Annual Meeting of the ACL*.

(Koller et al., 2009) A. Koller, K. Striegnitz, D. Byron, J. Cassell, R. Dale, S. Dalzel-Job, J. Oberlander, et J. Moore, 2009. Validating the web-based evaluation of NLG systems. Dans les actes de *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, Morristown, NJ, USA, 301–304. Association for Computational Linguistics.

(Kosseim et al., 2001) L. Kosseim, G. Lapalme, et S. Bauregard, 2001. Using information extraction and natural language generation to answer e-mail. *Data & Knowledge Engineering 38*(1), 85–100.

(Krahmer et Theune, 2010) E. Krahmer et M. Theune, 2010. *Empirical Methods in Natural Language Generation* (Lecture No ed.). New York, New York, USA : Springer Verlag.

(Lafferty et al., 2001) J. Lafferty, A. McCallum, et F. Pereira, 2001. Conditional random fields : Probabilistic models for segmenting and labeling sequence data. Dans les actes de *Proceedings of the Eighteenth International Conference on Machine Learning*, 282–289. Citeseer.

(Lamontagne et Lapalme, 2002) L. Lamontagne et G. Lapalme, 2002. Raisonnement à base de cas textuels–état de l'art et perspectives. *Revue de l'intelligence artificielle 16*(3), 339–366.

(Langkilde, 2000) I. Langkilde, 2000. Forest-based statistical sentence generation. Dans les actes de *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, 170–177. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA.

(Langkilde et Knight, 1998a) I. Langkilde et K. Knight, 1998a. Generation that exploits corpus-based statistical knowledge. Dans les actes de *Proceedings of the 36th annual meeting on Association for Computational Linguistics -*, Morristown, NJ, USA, 704. Association for Computational Linguistics.

(Langkilde et Knight, 1998b) I. Langkilde et K. Knight, 1998b. The practical value of n-grams in generation. Dans les actes de *In International Natural Language Generation Workshop*, 248–255.

(Langkilde-Geary, 2002) I. Langkilde-Geary, 2002. An empirical verification of coverage and correctness for a general-purpose sentence generator. Dans les actes de *Proceedings of the 12th International Natural Language Generation Workshop*, 17–24.

(Langus et Nespo, 2010) A. Langus et M. Nespo, 2010. SVO universally ? Experimental evidence for underlying word order and its consequences for language acquisition. Dans les actes de *Tokyo Conference on Psycholinguistics [TCP]*, Numéro 1995.

(Lapata, 2003) M. Lapata, 2003. Probabilistic text structuring : Experiments with sentence ordering. Dans les actes de *Proceedings of the 41st Annual Meeting on Association Of Computational Linguistic*, 545–552. Association for Computational Linguistics.

(Lavoie et Rambow, 1997) B. Lavoie et O. Rambow, 1997. A fast and portable realizer for text generation systems. Dans les actes de *Proceedings of the fifth conference on Applied natural language processing*, Numéro 1, 265–268. ACL.

(Lieberman et al., 1972) P. Lieberman, E. Crelin, et D. Klatt, 1972. Phonetic ability and related anatomy of the newborn and adult human, Neanderthal man, and the chimpanzee. *American Anthropologist 74*.

(Lin, 2004) C.-Y. Lin, 2004. Rouge : A Package for Automatic Evaluation of Summaries. *Proceedings of the Workshop on Text Summarization* (1).

(Loupy et al., 2010) C. D. Loupy, C. Ayache, S. Seng, et J.-m. T. Moreno, 2010. A French Human Reference Corpus for Multi-Document Summarization and Sentence Compression. Dans les actes de *LREC*, Numéro 1, Malta, 3113–3118.

(Marciniak et Strube, 2005) T. Marciniak et M. Strube, 2005. Using an annotated corpus as a knowledge source for language generation. Dans les actes de *Proceedings of UCNLG'05*, 19–24.

(Matthews, 1962) G. Matthews, 1962. Analysis by synthesis of sentences of natural languages. Dans les actes de *Proceedings International Congress on Machine Translation and Applied Language Analysis*, Numéro September, 5–8.

(Matthiessen et Bateman, 1991) C. M. I. M. Matthiessen et J. A. Bateman, 1991. *Text Generation and Systemic-Functional Linguistics : Experiences from English and Japanese*.

(McCallum et Li, 2003) A. McCallum et W. Li, 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. Dans les actes de *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 -*, Morristown, NJ, USA, 188–191. Association for Computational Linguistics.

(Mel'čuk, 1988) I. Mel'čuk, 1988. *Dependency syntax : theory and practice*. New York : State University of New York Press.

(Mel'čuk et A, 1970) I. Mel'čuk et v. A, 1970. Towards a Functioning Meaning-Text Model of Language. *Linguistics 57*, 10–47.

(Meunier et Danlos, 1998) F. Meunier et L. Danlos, 1998. FLAUBERT : user-friendly, multilingual NLG. Dans les actes de *INLG*, 284–287.

(Minnen et al., 2001) G. Minnen, J. Carroll, et D. Pearce, 2001. Applied morphological processing of English. *Natural Language Engineering 7*(03), 207–223.

(Nasr et al., 2004) A. Nasr, F. Béchet, et A. Volanschi, 2004. Tagging with hidden Markov models using ambiguous tags. Dans les actes de *Proceedings of the 20th international conference on Computational Linguistics*, Morristown, NJ, USA, 569. Association for Computational Linguistics.

(Nocera et al., 2002) P. Nocera, G. Linares, D. Massonié, et L. Lefort, 2002. Phoneme lattice based a• search algorithm for speech recognition. Dans les actes de *Text, Speech and Dialogue*, 83–111. Springer.

(Nothman et al., 2009) J. Nothman, T. Murphy, et J. Curran, 2009. Analysing Wikipedia and gold-standard corpora for NER training. Dans les actes de *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, Numéro April, 612–620. Association for Computational Linguistics.

(Ouattara et al., 2009) K. Ouattara, A. Lemasson, et K. Zuberbühler, 2009. Campbell's monkeys use affixation to alter call meaning. *PloS one 4*(11), e7808.

(Papineni et al., 2002) K. Papineni, S. Roukos, T. Ward, et W. Zhu, 2002. BLEU : a method for automatic evaluation of machine translation. Dans les actes de *Proceedings of the 40th annual meeting on association for computational linguistics*, Numéro July, 311–318. Association for Computational Linguistics.

(Paroubek, 2009) P. Paroubek, 2009. *Evaluating part-of-speech tagging and parsing* (Evaluation ed.)., Chapter Evaluating, 99–124. Numéro Evaluation of Text and Speech Systems. SpringerLink.

(Pinker et Jackendoff, 2005) S. Pinker et R. Jackendoff, 2005. The faculty of language : what's special about it ? *Cognition 95*(2), 201–36.

(Power, 2000) R. Power, 2000. Planning texts by constraint satisfaction. Dans les actes de *Proceedings of COLING 2000*, 642–648.

(Radev et McKeown, 1998) D. Radev et K. McKeown, 1998. Generating natural language summaries from multiple on-line sources. *Computational Linguistics 24*(3), 469–500.

(Rambow et Korelsky, 1992) O. Rambow et T. Korelsky, 1992. Applied text generation. Dans les actes de *Proceedings of the third conference on Applied natural language processing*, 40–47. Association for Computational Linguistics.

(Ratinov et Roth, 2009) L. Ratinov et D. Roth, 2009. Design Challenges and Misconceptions in Named Entity Recognition. Dans les actes de *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*. International Conference On Computational Linguistics.

(Raymond et Riccardi, 2007) C. Raymond et G. Riccardi, 2007. Generative and discriminative algorithms for spoken language understanding. Dans les actes de *Proceedings of Interspeech2007, Antwerp, Belgium*, 2. Citeseer.

(Reiter, 1994) E. Reiter, 1994. Has a consensus NL generation architecture appeared, and is it psycholinguistically plausible. Dans les actes de *Proceedings of the Seventh International Workshop on Natural Language Generation*, 163–170.

(Reiter, 1995) E. Reiter, 1995. NLG vs. templates. Dans les actes de *Proceedings of the Fifth European Workshop on Natural Language Generation*, Numéro August 1995, 95–105. Citeseer.

(Reiter et Belz, 2009) E. Reiter et A. Belz, 2009. An Investigation into the Validity of Some Metrics for Automatically Evaluating Natural Language Generation Systems. *Computational Linguistics 35*(4), 529–558.

(Reiter et Dale, 1997) E. Reiter et R. Dale, 1997. Building applied natural language generation systems. *Natural Language Engineering 3*(01), 57–87.

(Reiter et Dale, 2000) E. Reiter et R. Dale, 2000. *Building Natural Language Generation Systems*. Cambridge University. Press.

(Reiter et Gatt, 2009) E. Reiter et A. Gatt, 2009. SimpleNLG : A realisation engine for practical applications. Dans les actes de *Proceedings of the 12th European Workshop on Natural Language Generation*, 90–93. Association for Computational Linguistics.

(Riede et al., 2006) T. Riede, E. Bronson, H. Hatzikirou, et K. Zuberbuhler, 2006. Multiple discontinuities in nonhuman vocal tracts – A reply. *Journal of Human Evolution 50*(2), 222–225.

(Sabah et Zock, 1992) G. Sabah et M. Zock, 1992. La génération automatique de textes : trente ans déjà, ou presque. *Langages 26*.

(Saggion et al., 2002) H. Saggion, S. Teufel, D. Radev, et W. Lam, 2002. Meta-evaluation of summaries in a cross-lingual environment using content-based metrics. *Proceedings of the 19th*.

(Saussure et al., 1916) F. Saussure, C. Bally, A. Séchehaye, et A. Riedlinger, 1916. *Cours de linguistique générale* (2005 ed.). Payot.

(Schmid, 1994) H. Schmid, 1994. Probabilistic part-of-speech tagging using decision trees. Dans les actes de *Proceedings of International Conference on New Methods in Language Processing*, Volume 12. Manchester, UK.

(Scott et Moore, 2007) D. Scott et J. Moore, 2007. An NLG evaluation competition ? Eight reasons to be cautious. Dans les actes de *Proceedings of the Workshop on Shared Tasks and Comparative Evaluation in Natural Language Generation*, 22–23. Citeseer.

(Shannon, 1948) C. E. Shannon, 1948. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review 5*(1), 3.

(Shaw, 2002a) J. Shaw, 2002a. A corpus-based analysis for the ordering of clause aggregation operators. Dans les actes de *COLING02, Proceedings of the 19th International Conference on Computational Linguistics*, Morristown, NJ, USA, 1–7. Association for Computational Linguistics.

(Shaw, 2002b) J. Shaw, 2002b. *Clause Aggregation : An approach to generating concise text*. Thèse de Doctorat.

(Stolcke, 2002) A. Stolcke, 2002. SRILM-an extensible language modeling toolkit. Dans les actes de *Seventh International Conference on Spoken Language Processing*, Volume 3.

(Stone et Doran, 1997) M. Stone et C. Doran, 1997. Sentence planning as description using tree adjoining grammar. *Proceedings of the 35th annual meeting on Association for Computational Linguistics -*, 198–205.

(Stoyanov et al., 2010) V. Stoyanov, S. Processing, M. Baltimore, C. Cardie, et NY, 2010. Coreference Resolution with Reconcile. *ACL*.

(Sudoh et al., 2006) K. Sudoh, H. Tsukada, et H. Isozaki, 2006. Incorporating speech recognition confidence into discriminative named entity recognition of speech data. *ACL Proceeding*(July), 121–128.

(Surdeanu et al., 2008) M. Surdeanu, R. Johansson, A. Meyers, et L, 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. *Proceedings of the*, 159.

(Thompson, 1977) H. Thompson, 1977. Strategy and Tactics : a Model for Language Production. Dans les actes de *Papers from the 13th Regional Meeting. Chicago Ling. Soc. Chicago, Ill.*, Volume 13, Chicago Linguistics Society, 651–668.

(Tjong et Meulder, 2003) E. Tjong et F. D. Meulder, 2003. Introduction to the conll-2003 shared task : Language-independent named entity recognition. Dans les actes de *In CoNLL*.

(Turing, 1950) A. M. Turing, 1950. Computing Machinery and Intelligence. *Mind LIX*(236), 433–460.

(Varges et Mellish, 2001) S. Varges et C. Mellish, 2001. Instance-based natural language generation. Dans les actes de *ACL*.

(White, 2004) M. White, 2004. Reining in CCG chart realization. Dans les actes de *INLG 2004*, 182–191. Springer.

(Yip, 2006) M. J. Yip, 2006. The search for phonology in other species. *Trends in cognitive sciences 10*(10), 442–6.

(Yngve, 1961) V. Yngve, 1961. *Random generation of English sentences*. Numéro September. Massachusetts Inst. of Technology.

(Zouaq et Gagnon, 2010) A. Zouaq et M. Gagnon, 2010. Semantic Analysis using Dependency-based Grammars and Upper-Level Ontologies. Dans les actes de *CICLING 2010*. CICLING.