Chapitre 12
Génération Automatique de Textes

12.1. Introduction

La génération automatique de textes (GAT) est la branche du taln dont le but est de produire des énoncés en langage naturel à partir de représentations informatisées. Ceux-ci doivent être grammaticalement corrects, sémantiquement cohérents et pragmatiquement pertinents. La génération assure donc la fonction *émettrice* de la communication homme-machine, et se présente globalement comme le processus réciproque de la compréhension automatique (il s'agit là *d'exprimer* au lieu de *comprendre*). Cette comparaison tient principalement par l'identification possible des points d'entrée et de sortie des deux mécanismes□ l'attendu en sortie d'un générateur est du même type que les données d'entrée d'un analyseur, il s'agit de productions linguistiques (*i.e.* usuellement des textes écrits)□ de même, le résultat d'une analyse s'apparente à l'entrée d'un système de génération, il s'agit d'une représentation conceptuelle suffisamment abstraite pour faire l'objet d'un raisonnement artificiel. Ce niveau abstrait de codage est très peu consensuel et hors de portée d'un consensus dans l'état actuel des connaissances. Cette absence de consensus est peu dommageable pour la compréhension□ chaque chercheur peut se permettre de "□bricoler□" son niveau de représentation abstrait en fonction de son domaine (*e.g.* constat d'accident de voiture) sans se soucier de sa portabilité. Il est déjà si difficile de développer un système qui détermine automatiquement si la responsabilité d'un accident de la circulation revient au conducteur A ou B que l'on ne se soucie guère au premier chef de la généricité du niveau de raisonnement. Par contre, l'absence de consensus sur le niveau conceptuel est préjudiciable en génération à au moins deux titres□

– n'ayant pas un niveau d'entrées stable, les systèmes de génération sont difficilement comparables. On le sait, une absence de comparaison, qui entraîne une absence d'évaluation, ralentit la progression scientifique. De plus, elle entrave une répartition des tâches entre deux (ou plusieurs) équipes développant des générateurs dont les points d'interface ne sont pas équivalents ([Danlos 1999]).

– les applications de génération qui demandent à l'usager de fournir l'entrée dans un langage abstrait (*cf.* Section 12.2.1.) s'avèrent rédhibitoires, ce que l'on comprend. Il en résulte qu'il existe beaucoup moins d'applications industrielles de la

génération qu'il en existe de la compréhension. Cette disparité est accentuée par le fait que la compréhension de textes se limite souvent à l'analyse syntaxique de textes/phrases : la sortie de tels systèmes se réduit à la suite de représentations (morpho)syntaxiques des phrases du texte traitées isolément sans qu'aucun embryon de sens ne soit présent. Cette sortie est "acceptable" pour des applications comme la correction orthographique, par exemple. Or il n'existe pas d'application de "génération syntaxique" dont l'entrée serait la suite des représentations syntaxiques des phrases à générer (hormis la traduction automatique dans un système à transfert "de bas niveau", cf. Section 12.2.1).

Ces difficultés rencontrées en GAT sont cependant contrebalancées par un atout majeur : les applications de GAT n'ont pas besoin d'une couverture extensive de la langue tant du point de vue lexical que grammatical. En effet, ces applications, qui portent nécessairement sur un domaine donné (constat d'accident, bulletin météorologique, rapport boursier) dans un genre donné (récit, manuel, dialogue), peuvent naturellement (*i.e.* sans paraître monotones et en passant éventuellement le test de Türing) couvrir un sous-ensemble du lexique et de la grammaire. Ainsi une application de GAT dans le domaine de l'aéronautique n'aura pas besoin d'enregistrer le verbe *ressemeler* (réservé au domaine de la chaussure) ni le sens de *voler* synonyme de *dérober*. De plus, s'il s'agit d'instructions de maintenance, on écartera d'emblée les passifs impersonnels réservés à un genre plus littéraire. Ceci constitue un atout majeur : on sait que la pierre d'achoppement des systèmes d'analyse / compréhension provient du fait que l'usager jouit d'une totale liberté d'expression, sans avoir à contraindre son vocabulaire ni ses tournures grammaticales. D'où la construction de ressources énormes (*cf.* ce volume chapitres 4–8), et le problème de stockage et d'accès à ces ressources.

En résumé, la GAT se heurte au paradoxe suivant : c'est une technologie "mûre" pour les applications industrielles mais qui ne débouchent guère sur un vaste marché faute d'entrées stables. La Section 12.2 décrit les différents types d'application de la GAT et leurs apports par rapport au traitement de l'information. La Section 12.3 aborde l'architecture basique des systèmes de GAT, les tâches effectuées par les différents modules et les formalismes couramment utilisés. La Section 12.4 conclura sur les acquis et perspectives futures.

12.2. Pourquoi la génération automatique ?

La section suivante passe en revue les principales applications dans lesquelles intervient un module de génération. Pour chacune de ces applications, on s'attardera sur la question *Quoi-Dire ?*, *i.e.* sur la détermination du contenu informatif du texte généré. En effet, la production de textes demande de résoudre, d'une part, la question *Quoi-Dire ?*, ce qui débouche généralement sur une représentation conceptuelle des informations à transmettre, puis de résoudre la question *Comment-le-Dire ?*, ce qui revient à traduire cette représentation conceptuelle en un texte. Cette modularisation du processus de génération, qui conduit à une architecture en "pipe-line", sera décrite et critiquée dans la Section 12.3. Elle sera admise dans cette section où les

modules *Quoi-Dire* (“génération profonde”) et *Comment-le-Dire* (“génération de surface” abusivement abrégée en “génération” dans cette section) seront clairement dissociés.

Pour chacune des applications décrites, on citera quelques exemples de réalisation, sans prétendre à une quelconque exhaustivité (cf. [Zock 1992, Paiva 1998]). Après avoir passé en revue les applications de la GAT, nous serons en mesure de présenter ses apports dans le traitement de l’information (Section 12.2.2).

12.2.1. Types d’application

La première application historiquement développée en TALN est la TA(O) qui intègre un module de génération (en langue cible) quelle que soit l’architecture choisie. On rappellera qu’il existe deux architectures principales (*cf.* ce volume, chapitre 13) : l’architecture par transfert entre les représentations du texte source et du texte cible (ces représentations variant d’un niveau (morpho-)syntaxique peu abstrait à un niveau sémantico-conceptuel plus abstrait), voir Figure 12.1a, et l’architecture par pivot avec une seule représentation, nécessairement abstraite, commune aux textes source et cible, voir Figure 12.1b. Le module de génération dans un système à pivot doit résoudre tous les problèmes de la GAT, sauf celui de la question *Quoi-Dire*?, le contenu informatif du texte cible devant en principe être identique au contenu informatif du texte source. Le module de génération dans un système à transfert ne résout qu’un sous-ensemble des questions de la GAT : par exemple, il ne traite pas la question du découpage du texte en phrases, la structure du texte cible étant en principe identique à celle du texte source dans un système à transfert.

Figure 12.1. *Architectures de systèmes de TA(O) (a) par transfert, (b) par pivot*

Les systèmes de communication homme-machine intelligente comportent généralement un module de génération qui produit les réponses de la machine, voir Figure 12.2a. Dans ces systèmes, qui sont réalisés dans le cadre de dialogues (écrits ou oraux), d’interrogation de bases de données, d’interfaces en LN (*cf.* ce volume, chapitre 15), c’est un module de raisonnement (“système expert”) qui détermine le contenu de la réponse de la machine par exemple sous forme de buts communicatifs (*cf.* section 12.3.2.1). L’exemple le plus frappant de dialogue homme-machine est le système artimis développé au CNET [Panaget 1998]. Ce système est capable de gérer une quinzaine d’échanges oraux entre l’homme et la machine (l’extension parole est obtenue grâce à l’ajout de modules de reconnaissance et de synthèse vocales).

Pour résumer automatiquement un texte, il existe deux grandes méthodes : la première consiste à extraire des phrases du texte sur la base de calculs probabilistes (*cf.* ce volume, chapitre 11), l’autre simule l’activité humaine. Seule la seconde comporte un module de génération. La question *Quoi-Dire*? revient à extraire les

informations essentielles du texte. Si les réalisations industrielles de la première méthode abondent (malgré des résultats plutôt médiocres), seuls quelques prototypes de véritables résumés automatiques sont à l'étude ([Saggion 1999]).

Les rapports sur les données numériques (données météorologiques, boursières, statistiques) permettent de produire automatiquement de la documentation qui ne serait pas produite manuellement faute de temps ou de personnel pour la rédiger. La question *Quoi-Dire*? revient à mettre en avant les variations numériques importantes et éventuellement leurs corrélations, Figure 12.2b. La météo est un domaine prisé en GAT, comme en TA(O). fog pour la météo canadienne, développé par la société COGENTEX ([Kittredge 1991]), et multimeteo, pour les météos françaises, espagnoles, belges et autrichiennes, développé par la société ERLI/LEXIQUEST ([Coch 1998a]) sont deux applications multilingues notoires de la GAT. Dans le domaine statistique, le système PostGraphe ([Fasciano 1996]) est particulièrement novateur dans la mesure où il produit, à partir d'un tableau numérique, non seulement un texte mais aussi un graphique. Le choix d'un type de graphique (courbe, histogramme, secteurs *etc*.) se fait en fonction des données que l'on veut mettre en relief. Le texte souligne des informations peu lisibles sur le graphique.

Figure 12.2 *Systèmes (a) de dialogues, (b) de commentaires de données numériques*

Comme les rapports sur les données numériques, la production de manuels d'instructions permet d'offrir une large documentation. Cette application est donc prisée en GAT ([Kosseim 1995], drafter [Moore 1993]), bien qu'elle souffre d'un défaut majeur : c'est l'usager qui doit fournir manuellement l'entrée du système dans un langage abstrait. Même s'il est aidé d'outils pour effectuer cette tâche (*e.g.* menus déroulants contextuels), il doit maîtriser le langage abstrait, ce qui peut s'avérer rédhibitoire. On aurait pu penser, par exemple pour les manuels de logiciels, partir des spécifications du logiciel écrites en langage B, par exemple, et calculer le contenu du manuel à partir de ces spécifications. Malheureusement, dans la pratique, il ressort que les spécifications complètes ne sont pas disponibles (n'existent pas). Il en va de même pour les manuels de maintenance aéronautique. Le problème de fournir manuellement l'entrée du générateur se retrouve à l'identique pour d'autres types de documentation en masse : le courrier d'une entreprise, les rapports médicaux, les compte-rendus de réunions administratives sont autant de [2]domaines qui se prêtent bien à la GAT car ils sont suffisamment répétitifs sans être figés (pour des domaines totalement figés, *e.g.* courrier bancaire, la méthode des "phrases à trous" est suffisante, Section 12.2.3.). Ces domaines pourraient bénéficier des nombreux apports de la GAT décrits dans la section suivante. Il reste à surmonter l'obstacle de l'entrée du générateur. Dans cette perspective, [Power 1998] essaient de mettre au point un outil intelligent, wysiwym (*What You See Is What You Meant*) qui évite à l'usager de maîtriser le langage abstrait grâce à l'utilisation du langage naturel dans les menus déroulants.

Il existe encore d'autres applications mettant en jeu un module de génération (*e.g.* messages / diagnostics d'erreurs personnalisés dans les systèmes d'EAO, *cf.* [Zock 1994]), mais passons aux apports de la GAT.

12.2.2. Apports de la génération automatique

Un générateur explicite en langage naturel les informations contenues dans une base de données informatique□; c'est donc en quelque sorte un outil de documentation. Etant donné que la source d'informations à communiquer et le moteur de génération sont séparés, la GAT propose certains avantages liés, entre autres, à l'indépendance entre le stockage des informations et la réalisation de la documentation, et que l'on peut décliner comme suit□:

– *Réutilisibalité et maintenance*. Tout ajout ou modification d'informations se fait simplement dans la base de données, permettant la mise à jour automatique de la documentation□;

– *Recherche d'informations*. La recherche d'informations se fait aussi simplement dans la base de données, évitant ainsi la recherche d'informations textuelle aux résultats médiocres.

– *Flexibilité et interactivité*. A partir d'une même représentation abstraite un générateur peut proposer différentes paraphrases, en particulier dans le but d'améliorer le processus de communication.

– *Multilinguisme*. Un générateur multilingue peut produire des énoncés dans différentes langues à partir d'une même base de données de départ (variante de la paraphrase), en évitant le coût et les déboires de la traduction. Ceci est un atout majeur de la GAT car l'ajout d'une langue est peu coûteux.

– *Personnalisation et langues contrôlées*. Plus le système de génération est paramétrable, plus il est capable d'ajuster ses messages aux besoins de l'utilisateur ou aux circonstances applicatives□; en particulier, il peut produire des textes en langue contrôlée (et aussi vérifier que les consignes de rédaction en langue contrôlée sont cohérentes) [Lux 1998].

– *Sortie orale*. On obtient une sortie orale en interfaçant un générateur et un synthétiseur de la parole. L'interface permet de traiter efficacement la prosodie grâce aux annotations syntaxiques et lexicales du texte généré (*cf.* [Danlos 1986]).

– *Hypertexte*. La production d'hypertextes n'est techniquement pas plus difficile que celle de textes simples. Elle peut même alléger la tâche du générateur dans la mesure où un lien hypertexte laisse à l'usager le choix d'en savoir plus, sans que le système ait à déterminer la granularité des informations à transmettre.

– *Multimédia*. La production multimédia liant harmonieusement textes, sons et images est hautement attendue. Néanmoins, ce domaine en est à ses balbutiements car les problèmes théoriques sont nombreux.

– *Apport théorique*. La réflexion en GAT permet de s'interroger sur les mécanismes de production de la parole et donc sur le fonctionnement du langage. Elle permet donc des apports fructueux sur la langue et en particulier sur le discours (texte) trop largement ignoré dans les études de syntaxe cantonnées à la phrase prise

isolément.

12.2.3. Alternatives à la génération linguistique

Il existe des techniques de production de textes qui se contentent de manipuler des chaînes de caractères de manière opaque et ne font pratiquement pas intervenir de savoir linguistique. Ces méthodes n'offrent aucun des apports cités précédemment et pourtant, comme le fait remarquer [Reiter 1995], sur les centaines de milliers de logiciels dans le monde qui produisent du texte, environ 99,9% utilisent ce type de méthode. Pour cette raison, avant de commencer à décrire les mécanismes de la génération par traitement linguistique (section 12.3), nous allons d'abord présenter deux techniques rudimentaires (les messages pré-enregistrés et les formulaires) qui scientifiquement représentent la genèse de la GAT.

Messages pré-enregistrés (*canned messages*).

L'exemple typique d'utilisation de messages pré-enregistrés (*canned messages*, littéralement: "messages en conserve") est l'affichage de messages d'erreurs ou d'avertissements dans les logiciels informatiques. Le principe consiste à mettre en correspondance dans une table des codes, généralement numériques, (*e.g.* des codes d'erreurs) avec des textes (souvent brefs) stockés tel quels sous forme de chaînes de caractères. Lors de l'exécution du programme, si un événement renvoie un code remarquable, le message associé est alors affiché à l'écran. Ce type d'approche, technologiquement simple, ne fait intervenir aucune notion linguistique: les textes glosent une fois pour toutes les valeurs de certaines variables manipulées pas le programme, et cette correspondance tient lieu de relation de signification. Cette technique mérite à peine l'appellation "génération automatique" ou alors il convient de parler du degré zéro de la génération.

Formulaires (*templates*).

Les formulaires, encore appelés "textes à trous", représentent une première étape dans le perfectionnement de la technique des messages pré-enregistrés. Un formulaire est un texte partiellement rédigé (un *patron*), et les parties manquantes (les trous) sont remplacées par des variables littérales liées à des champs d'une base de données. Cette méthode est couramment utilisée pour la production massive de textes très répétitifs et ne variant que sur des éléments locaux, comme par exemple des lettres types pour le courrier administratif ou de vente par correspondance. L'atout des formulaires par rapport aux messages pré-enregistrés est qu'ils offrent plus de souplesse dans la personnalisation du texte et éventuellement la possibilité d'introduire quelques règles grammaticales superficielles. Cependant, les patrons étant figés par nature et le mécanisme d'instanciation des variables étroitement lié à la définition des patrons, les formulaires ne garantissent nullement la qualité linguistique des textes produits ni la maintenance aisée du système (*cf.* critiques dans [Reiter 1995, Coch 1998b]).

Comment la génération automatique ?

L'objet de la génération automatique par traitement linguistique étant de produire des énoncés en langue naturelle à partir de données informatiques, l'office d'un générateur, à savoir "assurer le passage de l'information au texte", est donc une affaire de transformation, ou plus exactement de transformation*s* car la production d'un texte est techniquement menée à bien en plusieurs étapes. Nous allons d'abord présenter dans cette section quels sont les grands types de données (ou de représentations des données) manipulés successivement dans le traitement et quels types de tâches ou d'opérations permettent de passer d'une représentation à une autre.

Architectures standards

12.3.1.1. Un traitement modulaire (architecture bipartite)

Il est clair qu'un système complet de GAT doit être capable de répondre à deux questions majeures : étant donné un ensemble de connaissances à la disposition d'un système informatique, *i)* quelles sont les informations à transmettre ? et *ii)* comment formuler ces informations abstraites en langue naturelle ? Points de mire de l'opérationnalisation, ces questions ont naturellement conduit à architecturer les systèmes de génération en deux grands modules séparés et successifs, qui reçoivent habituellement les appellations de *Quoi-Dire, composant stratégique, générateur profond*, ou *conceptualiseur* pour le premier, et *Comment-le-Dire, composant tactique, générateur de surface* ou *formulateur* pour le second (*cf.* Figure 12.3a).

Figure 12.3. *Architectures (a) bipartite, (b) tripartite de système de GAT*

Le *Quoi-Dire* a donc pour fonction d'élaborer la *signification* du texte et pour ce faire il a recours à des ressources variées qui comprennent des bases de connaissances génériques et/ou spécifiques à un domaine, des données encyclopédiques, des règles de pragmatique, des modèles de locuteur, des mécanismes d'inférences logiques, *etc.* Le *Comment-le-Dire* est lui chargé de "l'habillage linguistique" du message ; les ressources qu'il utilise sont des connaissances sur la langue, *i.e.* principalement un lexique et une grammaire.

Il se trouve que cette division bipartite ne répond pas uniquement à des préoccupations d'ordre technologique, mais qu'elle a pu aussi recevoir une caution psycholinguistique (*cf.* en particulier la modélisation du locuteur de [Levelt 1989]).

12.3.1.2. Critique (architecture tripartite vs. intégrée)

Quoi-Dire et *Comment-le-Dire* figurent les deux axes principaux du cahier des charges de tout système de génération ; c'est-à-dire qu'un générateur doit minimalement savoir quoi dire et comment le dire. Mais il ne va pas de soi que ces deux objectifs se projettent nécessairement sur un découpage similaire en tâches ou processus : les décisions de haut niveau qui manipulent des entités d'information abstraites comme des unités et des relations sémantiques ou des concepts organisés sont censées être localisées à la hauteur du *Quoi-Dire*, or certaines de ces décisions

peuvent très bien relever de la démarche du *Comment-le-Dire*. En effet, fonctionnellement, dès lors qu'il s'agit d'organiser, d'agencer ou de planifier, d'une certaine manière on commence à répondre à "comment le dire?".

Depuis quelques années, on voit émerger des propositions de "méta-architectures" consensuelles plausibles formulées en termes plus techniques ([Reiter 1994, Cahill 1999]). Le modèle qui en ressort s'articule plutôt en trois étapes, identifiées comme: *Détermination de Contenu (DC)*, *Planification de Phrase (PP)* et *Réalisation de Surface (RS)* (*cf.* Figure 12.3b). Ce type d'architecture est *de facto* générique car il est issu d'une étude comparée de plusieurs systèmes opérationnels existants; il rend compte davantage de ce qui se fait (ou s'est fait) que de ce qui devrait se faire. Mais, et même s'il se superpose assez naturellement au modèle bipartite (*DC* correspondant à *Quoi-Dire*, *PP* et *RS* à *Comment-le-Dire*), il présente l'intérêt d'expliciter un peu plus clairement la nature opératoire des composants et les points d'interfaçage entre ces composants.

De plus, les faits montrent que les systèmes existants fonctionnent en mode *pipeline*, c'est-à-dire que pendant le traitement, les différents états du message en cours d'élaboration se succèdent suivant un flux unidirectionnel d'un module à l'autre sans retour en arrière possible vers un module supérieur. Une architecture modulaire en pipeline offre une grande souplesse pour le développement, la maintenance et éventuellement l'exportabilité du système. Cependant, dès le milieu des années 1980, plusieurs études (*cf.* [Appelt 1985, Danlos 1985, Rubinoff 1992, De Smedt 1996]) ont dénoncé l'inadéquation et l'arbitraire du traitement en pipeline en insistant sur les interactions possibles certains choix stratégiques de haut niveau et les réalisations de surfaces. Ces critiques ont souvent donné lieu à des propositions d'architectures intégrées où les modules sont interconnectés ou interdépendants. Malgré cela, force est de constater qu'une majorité des systèmes de génération existant ont été développés en privilégiant la concession et le compromis du pipeline. Sans aller jusqu'à parler de schisme, ce point témoigne des écarts pouvant parfois séparer les préoccupations de la recherche théorique de celles de la mise en œuvre technique.

12.3.2. Les mécanismes de la génération automatique

Dans les sections suivantes, nous allons décrire plus précisément les différents types de tâches auxquelles procèdent les modules composant les architectures présentés dans la section précédente. Autant que possible nous essaierons d'indiquer la nature des entrées et sorties types de chaque composant. Malgré l'esquisse de consensus évoquée plus haut, l'ordre dans lequel sont présentées les tâches ne correspond pas forcément aux enchaînements que l'on peut rencontrer dans les systèmes existants, d'autant plus qu'il est courant de voir des systèmes qui fusionnent plusieurs tâches dans un seul composant et donc souvent un seul processus. Ainsi, 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 mentionné plus haut, encore nommé *micro-planificateur* ou *module linguistique* et dont l'objet est de mettre en place le matériau linguistique qui va "□ncarner□ le message, contient communément les tâches de *Planification Syntaxique* et *Lexicalisation*.

12.3.2.1. Sélection du Contenu Profond (SCP).

Entrée□ bases de connaissances informatiques ou buts communicatifs
Sortie□ réseaux conceptuels/sémantiques

Cœur même du *Quoi-Dire* (comme son nom l'indique), cette tâche intervient en amont du traitement. Son rôle consiste à fournir une *spécification du contenu sémantique* du texte à générer. La *SCP* mets en œuvre des opération de sélection d'informations et une opération de transcription de ces informations en réseaux ou représentations sémantiques, conceptuelles ou logiques… Comme la *SCP* intervient dès le début du traitement et que, pratiquement, les types de données exploitées en entrée sont aussi divers que les systèmes de génération sont nombreux, les méthodes employées sont nécessairement très variées. Nous distinguerons ici trois grands types d'approches (en précisant que des systèmes peuvent combiner ces approches).

Tout d'abord, rappelons (*cf.* section 12.2.1) que certains générateurs confient, partiellement ou complètement, la *SCP* à la charge de l'utilisateur (par l'intermédiaire d'une interface qui permet de saisir un contenu), ou à une application externe au générateur (système expert, planificateur de tâches, *etc.*). Dans ce cas, le rôle du module se limite principalement à convertir (et éventuellement à organiser) les données présélectionnées en une structure sémantique ou conceptuelle qui prépare la contexture du message linguistique. Dans ce type d'approche, la *SCP* se fait souvent conjointement à la structuration rhétorique (*cf.* plus bas).

D'autres générateurs prennent en entrée des buts communicatifs. Un but communicatif est généralement modélisé comme un état mental de l'utilisateur (*i.e.* l'interlocuteur) que le générateur cherche à atteindre par le truchement de son acte de langage. Ces états représentent le plus souvent une connaissance ou une compétence□ par exemple, l'état *(competent ?agent (do ?agent ?act))* signifie que l'utilisateur (*?agent*) dispose des informations nécessaires pour exécuter l'action *?act*, l'état *(persuaded□agent□p)* que l'utilisateur est convaincu de la validité de la proposition *?p* (exemples tirés de [Moore 1993]). La sélection du contenu se fait alors par "□déploiement□" des buts communicatifs□ à partir d'un but donné, le système puise dans sa base de connaissances en quels sous-buts il peut se décomposer, c'est-à-dire quels sous-buts satisfont le but communicatif initial. Chaque but apporte un complément d'information et est ensuite converti en acte de langage (comme affirmer, conseiller, ordonner, expliquer… la proposition *?p*).

La troisième catégorie de générateurs part d'un fonds de données ou de connaissances renseignant assez largement un domaine. Dans cette approche, le système peut aussi manipuler des buts communicatifs, mais ils ne guident pas la sélection du contenu de la même manière. En règle générale, il s'agit de buts plus vagues et plus globaux, comme "□décrire un objet, un fait□ ou plus simplement "□ommuni-

quer tout ce que l'*on* sait sur un sujet particulier□ et la substance du message à générer est donc un sous-ensemble de la base de connaissances. La *SCP* doit alors raisonner afin de filtrer les informations pertinentes et/ou saillantes en s'appuyant, par exemple, sur des schémas prototypiques, des heuristiques, des règles empiriques, des principes coopératifs… Souvent il s'agit presque autant de calculer "□quoi dire□" que "□quoi ne pas dire□". En effet, un abondance de détails, surtout s'ils sont superflus, tendrait rapidement à noyer le propos et à nuire à la fluidité voire à la cohérence du texte. Des tentatives ont été faites pour paramétrer les seuils de la superfluité et de la pertinence en fonction d'un profil d'utilisateur afin de produire des messages adaptés.

L'opération de transcription vers une représentation sémantique prend toute son importance dans les systèmes dont l'entrée se présente sous forme de données numériques brutes (ou assimilées). C'est lors de cette étape que peuvent être inférées des entités conceptuelles comme□ augmentation de taux, variations de températures, déplacement d'un mobile sur une trajectoire, *etc.*

12.3.2.2. Structuration Rhétorique (SRh).

Entrée□ réseaux conceptuels/sémantiques ou buts communicatifs
Sortie□ plan de texte

Une structure sémantique, obtenue par la SCP, indique le *sens* global d'un texte, mais il apparaît de plus en plus qu'elle nécessite un traitement supplémentaire avant d'être injectée dans les composants linguistiques proprement dits (*Comment-le-Dire*). D'abord lorsque le contenu profond du message comporte un grand nombre d'informations et que le texte à générer atteint la dimension d'un ou plusieurs paragraphes, la structure sémantique devient un objet très complexe et touffu qui confrontera les modules syntaxiques et lexicaux (*cf.* plus bas) à une haute combinatoire de choix particulièrement dommageable pour l'efficacité du traitement. Par ailleurs, la structure sémantique ne rend pas compte du *cheminement discursif* c'est-à-dire dans quel ordre et sous quels rapports les "□idées□" sous-jacentes du texte vont se présenter au lecteur. La structuration rhétorique est donc la phase *d'organisation* du message dans un suivi textuel cohérent et fluide. Certains systèmes l'intègrent étroitement au module *SCP*, d'autres dissocient clairement les deux tâches en faisant intervenir la *SRh* après la *SCP*.

Techniquement, la *SRh* consiste en une opération de segmentation□/ regroupement du contenu sémantique en unités qui se réaliseront ensuite sous forme de phrases ou de propositions syntaxiques, et en une opération d'articulation de ces unités dans un plan de texte cohérent. L'articulation du plan est assurée par des relations dites *de discours* ou *rhétoriques* (le modèle le plus utilisé en génération est la RST, *cf.* section 12.3.4.). Elles jouent le rôle de "□liant de cohérence□" du texte d'une part en hiérarchisant les phrases ou propositions entre elles, et d'autre part en explicitant des rapports pragmatiques ou rhétoriques comme, par exemple, *l'explication, le résultat, l'élaboration, le but…* Dans la suite du traitement, la présence d'une relation rhétorique peut permettre de sélectionner un connecteur

discursif (conjonction de coordination, de subordination, adverbe…). Pour bien saisir la nature des relations rhétoriques, il faut les voir comme à la charnière des relations sémantiques et des constructions syntaxiques. Proches des relations sémantiques, elles ne relient cependant pas des unités de sens minimales mais plutôt des unités plus complexes à l'échelle de la proposition. Par ailleurs, elles annoncent d'une certaine manière la "syntaxe du texte", mais en se situant à un niveau hyperphrastique contrairement à la syntaxe traditionnelle (hypophrastique). Le choix d'une relation de discours se fait généralement par déduction à partir d'une ou plusieurs relation(s) sémantique(s) ou conceptuelle(s), mais comme l'ont montré [Kosseim 1995], la projection du niveau sémantique vers le niveau rhétorique n'est ni triviale ni unidirectionnelle; par exemple, une relation sémantique de *cause à effet* entre une opération et un état peut donner lieu à différents enchaînements rhétoriques:

– *Abaisser le levier A. Le circuit se met en marche.* (résultat)
– *Pour mettre le circuit en marche, abaisser le levier A.* (but)
– *Vous pouvez mettre le circuit en marche en abaissant le levier A.* (moyen)

12.3.2.3. Planification Syntaxique (PS)

Entrée: réseau sémantique ou plan de texte
Sortie: arbres syntaxiques

Cette tâche procède à la sélection des constructions grammaticales qui vont charpenter les phrases du texte. Il s'agit là de convertir une structure sémantique (éventuellement pré-organisée rhétoriquement) en arbre syntaxique et *stricto sensu*, la *PS* opère en projetant les relations sémantiques vers des relations ou fonctions syntaxiques. Par exemple, la relation de partie au tout (*Part-of*) peut donner lieu à un groupe prépositionnel en *de* pour le français, "le moteur *de* l'avion", à un génitif dans d'autre langues, "the aircraft*'s* engine". Il faut noter qu'un nombre important de systèmes de génération utilise des lexiques-grammaires intégrés ou des grammaires lexicalisées (par exemple les systèmes basés sur les TAGs, *cf.* section 12.3.4) dans lesquels les constructions syntaxiques sont ancrées par des entrées lexicales; la *PS* s'effectue alors conjointement à la lexicalisation.

12.3.2.4. Lexicalisation

Entrée: arbres syntaxiques profonds ou réseaux sémantiques
Sortie: arbres ou réseaux lexicalisés

La *Lexicalisation* est la tâche complémentaire de la *Planification Syntaxique* puisqu'elle consiste à choisir les mots de la langue qui devront "incarner" les sens préalablement déterminés. Ce choix porte typiquement sur les *mots pleins* (noms, verbes, adjectifs, adverbes) par opposition aux *mots outils* (prépositions, déterminants, conjonctions…) qui eux peuvent relever de la PS. Comme nous venons de le mentionner, la *Lexicalisation* s'associe avantageusement à la *Planification Syntaxique*; en effet convertir une unité de sens ou un concept en item lexical entraîne de fait le choix d'une catégorie grammaticale et il est indispensable de veiller à la compatibilité syntaxique des catégories de tous les mots et syntagmes qui vont

"habiter" la phrase. Par exemple *branchement de l'appareil* pourra être argument d'un verbe, mais pas *brancher l'appareil*, pourtant sémantiquement équivalent.

12.3.2.5. Ajustement Morphologique.

Entrée : arbres syntaxiques
Sortie : séquences orthographiques linéaires ou arbres syntaxiques enrichis

Passé la Planification Syntaxique et la Lexicalisation, le message en cours de synthèse se présente sous la forme d'un arbre syntaxique dont les nœuds ou les feuilles sont instanciés par des items lexicaux, *i.e.* des entrées de dictionnaire. Pour obtenir la forme de surface correcte du texte, il reste encore, outre à linéariser l'arbre (ce qui est trivial à partir d'un arbre de constituants), à effectuer la flexion des mots issus du dictionnaire de l'application, c'est-à-dire : conjuguer les verbes, accorder les noms, les pronoms, les déterminants, les adjectifs… mais aussi agglutiner des séquence *préposition + déterminant* (*de + le du*), élider certains articles ou certaines prépositions (*le*, *la*, *de*, *ce* devant une voyelle deviennent *l'*, *d'*, *cet*), ajouter des traits d'union…

12.3.2.6. Formatage Typographique.

Entrée : arbres syntaxiques de surface ou chaînes de caractères
Sortie : fichiers de texte éventuellement formatés ou balisés

Le *Formatage Typographique* n'est pas une opération de haute technologie, mais elle est indispensable pour rendre le texte lisible à l'utilisateur humain. Elle se présente comme une étape de *finition* et elle opère principalement sur des chaînes de caractères en ajoutant les majuscules en début de phrase et sur les noms propres ou les sigles, en insérant la ponctuation (sans oublier les espaces entre les mots et les sauts de paragraphe), mais aussi dans certains cas en effectuant la mise en forme graphique (titres, italiques, caractères gras, listes à puces…). De plus en plus de systèmes produisent des textes codés dans des langages de composition de documents comme SGML/XML, HTML, LaTeX… plutôt que sous forme de fichiers ascii bruts : ils peuvent aussi alors insérer des images et/ou des liens hyper-textuels. Dans le cas de la génération de messages oraux, ce module est remplacé par un programme de synthèse vocale.

12.3.2.7. Autres tâches

Les six tâches que nous venons de présenter constituent la principale ligne directrice du traitement linguistique standard d'un générateur. Plusieurs autres types d'opérations peuvent venir se greffer sur cet ensemble, comme la conception et le choix d'actes de langages particuliers, le marquage de la topologie communicationnelle (en thème et rhème), l'adoption d'un style de langue subjectif, *etc*. Nous en exposerons ici deux des plus importantes, la *génération d'expressions référentielles* et l'*agrégation*.

Génération d'expressions référentielles ([Dale 1992a], [Danlos 1992]). Le problème que cherche à résoudre la génération d'expressions référentielles est : "comment faire référence sans ambiguïté à des concepts ou des entités ?". Cette opération

peut en quelque sorte être vue comme la composante chargée de planifier les *groupes nominaux* du texte du fait qu'elle traite les questions du choix de la détermination (*le* vs. *un* vs. *ce bouton*), du prédicat lexical (*l'appareil* vs. *le magnétoscope*), des éventuels types d'épithètes restrictifs (*le bouton de gauche* vs. *le bouton vert* vs. *le premier bouton* vs. *le bouton 'rwd'*…) et de l'emploi adéquat de formes pronominales (*il*/*le* vs. *l'appareil*).

Agrégation. Il est assez difficile de trouver une définition arrêtée de l'agrégation (*cf.* [Reape 1999]) malgré l'intérêt croissant qui est porté à cette notion. Tombent sous le chef d'agrégation toutes les opérations qui ont pour effet de rendre le texte plus concis, plus lisible, plus naturel, moins redondant… et qui synthétiquement consistent à regrouper plusieurs entités de la représentation conceptuelle, sémantique voire syntaxique en une seule entité plus globale. En surface, l'agrégation a pour conséquence une factorisation (au sens mathématique) des constituants du texte. Entre autres, cette opération permet de générer des syntagmes pluriels par simple sommation (en obtenant par exemple *les (deux) manettes* à partir de *manette*(x_1) *manette*(x_2)), par coordination partagée (*e.g.* l*es manettes L1 et L2*), par hyperonymie (*e.g. écran(x) clavier (y) périphérique({x,y})*), par un terme collectif (*e.g.* en anglais la séquence {*monday, tuesday,…, friday*} peut s'agréger en *weekdays*), *etc.* C'est aussi par agrégation que peuvent être obtenues des coordinations de constructions syntaxiques (*e.g. tirer puis relâcher la manette*) et des ellipses (*insérer la fiche 1 dans le port d'entrée et la fiche 2 dans la sortie*). La capacité de générer des textes "agrégés" est fondamentale pour garantir une réelle qualité linguistique, et il est frappant de constater que les techniques mises en œuvre dans les différents systèmes développés sont non seulement très diverses, mais qu'elles peuvent aussi intervenir à des niveaux très variés (de la représentation conceptuelle à la forme de surface).

12.3.4. Formalismes linguistiques

Les mécanismes de la GAT opèrent sur divers types de représentations du message et mettent en jeu un certain nombre de règles linguistiques. Ces notions sont le plus souvent héritées de formalismes linguistiques déjà développés dans un cadre théorique ou applicatif. Loin de toute tentative d'exhaustivité, nous terminerons en mentionnant quelques uns des formalismes les plus représentatifs, les plus exploités et les plus adaptés à la GAT.

– *Graphes Conceptuels*, ([Sowa 1984]). Il ne s'agit pas à proprement parler d'un formalisme linguistique mais plutôt d'un langage de représentation des connaissances dédié aux applications d'Intelligence Artificielle. Les graphes conceptuels se situent dans la lignée des ontologies formelles (Aristote, Pierce…) et des réseaux sémantiques. Ils s'appuient sur un système de notation graphiques dans lesquelles des nœuds conceptuels sont connectés par des nœuds relationnels. Conçus pour les implémentations informatiques, les graphes conceptuels sont depuis longtemps souvent utilisés dans les modules de *Quoi-Dire* parce qu'ils permettent des représentations sémantiques non hiérarchiques (contrairement à la logique prédicative) qui

permettent de déduire plusieurs variantes de structures linguistiques à partir d'une structure de sens donnée (*cf.* [Nogier 1991, Nicolov 1999]).

– *Théorie Sens-Texte* (TST). La Théorie Sens-Texte ([Mel'cuk 1988]) propose un modèle de représentations linguistiques particulièrement adapté au processus de la génération. Le modèle est stratifié sur sept niveaux de représentation : sémantique, syntaxe profonde, syntaxe de surface, morphologie profonde… jusqu'à phonologie de surface. La TST établit des jeux de règles permettant la conversion structurale et lexicale d'une représentation linguistique d'un niveau vers une représentation d'un niveau adjacent. Ces règles étant décrites de manière formelle, leur intégration dans un générateur offre un composant lexico-grammatical directement opérationnel (*cf.* gossip, fog, lfs [Kittredge 1991], [Polguère 1998], joyce [Rambow 1992]).

– *Grammaires d'Arbres Adjoints* (Tree Adjoining Grammar, TAG). Les TAG (*cf.* [Joshi 1987]), sont des grammaires arborescentes lexicalisées. Les constructions syntaxiques de la langue n'y sont pas représentées par des règles de réécriture mais par des arbres élémentaires de profondeur quelconque, déterminant ainsi des domaines de localités pertinent pour la syntaxe. La structure d'une phrase s'obtient en combinant plusieurs arbres, et la trace de la combinaison est sténographiée dans un arbre dit *de dérivation*, facilement assimilable à un graphe sémantique. Les arbres étant ancrés à des entrées du lexiques (et éventuellement aux concepts associés), ils assurent un lien rigoureux entre représentations abstraites et représentation lexico-grammaticales. Depuis le début des années 80, les TAG sont fréquemment utilisés en génération que se soit dans leur forme standard ou dans des versions adaptées (G-TAG [Danlos 1998] dans flaubert [Meunier 1997] et clef [Meunier 1999], DSG, *D-tree Substitution Grammar*, dans protector [Nicolov 1999]).

– *Théorie des Structures Rhétoriques* (*Rhetorical Structures Theory*, RST). La RST ([Mann 1987]) se présente comme un modèle générique de "grammaire" d'organisation des textes. La structure globale d'un texte consiste en une série de regroupements hiérarchiques des segments textuels effectués par l'intermédiaire de *schémas* qui recouvrent récursivement des empans de texte. Chaque schéma est défini par une relation rhétorique (Explication, Résultat, Elaboration…) qui spécifie sous quelles conditions deux segments peuvent être regroupés pour former un segment de niveau supérieur, et quel effet pragmatique provoque le regroupement. La grande majorité des générateurs pratiquant la *Structuration Rhétorique* s'appuient actuellement sur la RST (*cf.* par exemple [Hovy 1991]) malgré l'imprécision et la pauvreté formelle de la théorie.

– *Grammaire systémique*. Le cadre théorique de la linguistique systémique ([Halliday 1994]) s'oppose aux modèles de grammaires de constituants en ce sens qu'il place au cœur de sa description le processus de formation d'énoncé par le locuteur en termes de fonctionnalités ("que fait le langage…") plutôt qu'en termes de constitutions ("comment est fait le langage…"). Une grammaire systémique peut se présenter sous la forme d'un réseau de choix (système) dont le parcours rend compte de l'ensemble des décisions à prendre pour mener à bien la confection d'un message. Le caractère intrinsèquement procédural de ces grammaires s'est naturelle-

ment très tôt avéré approprié au développement de systèmes de génération (*cf.* komet-kpml [Teich 1994])

– *Grammaire d'unification fonctionnelle* (*Functional Unification Grammar, FUG*). Une des originalités du formalisme FUG ([Kay 1979]) a été de représenter les constructions syntaxiques et les données lexicales sous un même format de *matrices de traits* (appelées encore *structures fonctionnelles*). Partant, FUG ne propose pas en soi une théorie linguistique, mais un outil générique entièrement basé sur l'unification et qui permet de coder des grammaires pour les traitements informatiques. C'est dans ce formalisme qu'un interpréteur, fuf (*Functional Unification Formalism*), et une grammaire de l'anglais, surge, spécifiquement dédiés aux besoins de la GAT, ont été développés et mis à la disposition de la communauté, notamment par M. Elhadad (*cf. http://www.cs.bgu.ac.il/fuf/*)

12.4. Acquis et perspectives

Dans les acquis de la GAT, on signalera tout d'abord la réalisation d'un certain nombre de générateurs (multilingues) performants dans des domaines ciblés (la performance de ces systèmes est obtenue en partie par le fait qu'une couverture exhaustive de la langue n'est pas nécessaire). La performance peut aussi être obtenue par le fait que certains développements s'autorisent à hybrider leurs technologies en incorporant dans le générateur linguistique quelques composants statiques (messages pré-enregistrées ou formulaires) afin de recentrer l'effort scientifique sur les problèmes et les objectifs spécifiques de l'application et d'éviter ainsi d'avoir à traiter toutes les subtilités de la langue (*cf.* [Reiter 1995, Coch 1998b]). On voit aussi poindre des plate-formes multi-tâches, multi-utilisateurs, multi-domaines et multilingues (comme clef [Meunier 1999]). On rappellera cependant que la GAT offre une vaste panoplie d'applications qui restent non réalisées car leurs entrées ne peuvent être que manuelles donc rédhibitoires. Il reste à espérer que l'informatisation galopante des données permettra de contourner cet obstacle.

Un autre acquis est qu'il existe un embryon de consensus sur l'architecture d'un système de GAT, en tout cas sur la liste et la définition des tâches à effectuer. La modularisation idéale reste encore à trouver, mais on peut d'ores et déjà s'appuyer sur la prise de conscience généralisée de la difficulté à modulariser entre *Quoi-Dire* et *Comment-le-Dire*.

L'évaluation en GAT est difficile : comment définir précisément des critères pour répondre à "qu'est-ce qu'un bon générateur ?". Il ne suffit pas de statuer sur la qualité linguistique finale de la sortie, il faut aussi que le texte produit satisfasse au mieux les objectifs qui motivent le processus de génération. Une telle évaluation ne peut être qu'humaine (et donc subjective), des protocoles de tests précis doivent être établis. Rappelons également que la diversité inévitable des types d'entrées rend la confrontation de différents systèmes pour le moins hasardeuse. Néanmoins, certains efforts ont été entrepris pour amorcer une évaluation des systèmes de GAT.

Les recherches en GAT ont induit un effort important sur les phénomènes discursifs, entre autres, sur la cohérence et cohésion d'un texte et sur les relations et

connecteurs discursifs. Les questions syntaxiques sont réglées□ on sait bien générer des phrases (avec éventuellement des liens hypertextuels). Il reste à poursuivre les recherches sur le niveau textuel et à approfondir notre compréhension de l'activité de production de langage.

Bibliographie

[Adorni 1996] Adorni, G. et Zock, M., Eds. *Trends in Natural Language Generation. An Artificial Intelligence Perspective. Proceedings of the 4th European Workshop (ENLG'93), Pisa*. Springer-Verlag, 1996.

[Appelt 1985] Appelt, D. E. *Planning English Sentences*. Cambridge University Press, 1985.

[Cahill 1999] Cahill, L., Doran, C., Evans, R., Mellish, C., Paiva, D., Reape, M., Scott, D., et Tipper, N. "□In search of a reference architecture for NLG systems□". In *Proceedings of the 7th European Workshop on Natural Language Generation (ENLG'99)*, pp.□ 77–85 Toulouse, France, 1999.

[Coch 1998a] Coch, J. "□Interactive generation and knowledge administration in Multi-Meteo ". In *Proceedings of the 9th International Workshop on Natural Language Generation (INLG'98)*, pp.□300–303, Niagara-on-the-Lake, Ontario, 1998.

[Coch 1998b] Coch, J. "□Applications industrielles de le génération : pourquoi et comment□". *La génération de textes, t.a.l.*, 39(2), pp. 89–105, 1998.

[Dale 1990] Dale, R., Mellish, C. et Zock, M., Eds. *Current Research in Natural Language Generation*. Academic Press, Harcourt Brace Janovich, New York, 1990.

[Dale 1992a] Dale, R. *Generating Referring Expressions*. MIT Press, 1992.

[Dale 1992b] Dale, R., Hovy, E., Rösner, D. et Stock, O., Eds. *Aspects of Automated Natural Language Generation. Proceedings the 6th International Workshop on Natural Language Generation (INLG'92), Trento*. Springer-Verlag, 1992.

[Danlos 1985] Danlos, L. *Génération automatique de textes en langues naturelles*. Masson, Paris, 1985.

[Danlos 1986] Danlos, L., Emerard, F. et Laporte, E. "□Synthesis of spoken messages from semantic representation□". In *Proceedings of COLING'86*, Bonn, RFA, 1986.

[Danlos 1992] Danlos, L. "□Contraintes syntaxiques de pronominalisation en génération de textes□". *Langages*, 106, pp.□36–62, 1992.

[Danlos 1998] Danlos, L. "□G-TAG□ un formalisme lexicalisé pour la génération de textes inspiré de TAG□". *La génération de textes, t.a.l.*, 39(2), pp. 7–33, 1998.

[Danlos 1999] Danlos, L. et Lapalme, G. " An experiment in combining two text genera-

tors». In *Proceedings of AISB'99, Workshop on Reference Architectures and Data Standards for NLP*, Edinburg, 1999.

[De Smedt 1996] De Smedt, K., Horacek, H. et Zock, M. "Architectures for natural language generation: Problems and perspectives". In [Adorni 1996].

[Fasciano 1996] Fasciano, M. et Lapalme, G. "PostGraphe: a system for the generation of statistical graphics and text". In *Proceedings of the 8th International Workshop on Natural Language Generation (INLG'96)*, pp. 51–60, Herstmonceux, Sussex, UK, 1996.

[Halliday 1994] Halliday, M. A. K. *An Introduction to Functional Grammar*. Arnold, London, second edition, 1994.

[Hovy 1991] Hovy, E. H. "Approaches to the planning of coherent texts". In [Paris 1991].

[Joshi 1987] Joshi, A. K. "The relevance of tree adjoining grammar to generation". In [Kempen 1987].

[Kay 1979] Kay, M. "Functional grammars". In *Proceedings of the 5th Annual Meeting of Berkeley Linguistics Society*, pp. 142–158, Berkeley, 1979.

[Kempen 1987] Kempen, G., Ed. *Natural Language Generation. New results in artificial intelligence, psychology and linguistics*. Martinus Nijhoff Publisher, Dordrecht, 1987.

[Kittredge 1991] Kittredge, R. et Polguère, A. "Dependency grammars for bilingual text generation: Inside FoG's stratificational models". In *Proceedings of the International Conference on Current Issues in Computational Linguistics*, pp. 318–330, Penang, Malaisie, 1991.

[Kosseim 1995] Kosseim, L. et Lapalme, G. "Choosing rhetorical relations in instructional texts: The case of effects and guidance". In *Proceedings of the 5th European Workshop on Natural Language Generation (ENLG'95)*, Leiden, the Nederlands, 1995.

[Levelt 1989] Levelt, W. J. M. *Speaking – From Intention to Articulation*. MIT Press, 1989.

[Lux 1998] Lux, V. Français rationalisé étendu modulaire – Tests en génération automatique. Thèse de doctorat en linguistique, Université Paris 7, 1998.

[Mann 1987] Mann, W. C. et Thompson, S. A. "Rhetorical structure theory: description and construction of texts structures". In [Kempen 1987].

[McKeown 1985] McKeown, K. R. *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text*. Cambridge University Press, 1985.

[Mel'cuk 1988] Mel'cuk, I. *Dependency Syntax: Theory and Practice*. State University of New York Press, New York, 1988.

[Meunier 1997] Meunier, F. Implantation du formalisme de génération G-TAG. Thèse de doctorat en informatique, Université Paris, 1997.

[Meunier 1999] Meunier, F. "Modélisation des ressources linguistiques d'une application industrielle". In *Actes de TALN'99*, Cargèse, France, 1999.

[Moore 1993] Moore, J. D. et Paris, C. L. "Planning text for advisory dialogues:

capturing intentional and rhetorical information". *Computational Linguistics*, 19(4), 1993.

[Nicolov 1999] Nicolov, N. et Mellish, C. " Protector: Efficient sentence generation with lexicalised grammars ". In *Proceedings of the 7th European Workshop on Natural Language Generation (ENLG'99)*, pp.96–105, Toulouse, France, 1999.

[Nogier 1991] Nogier, J.-F. *Génération automatique de langage et graphes conceptuels*. Hermes, Paris, 1991.

[Paiva, 1998] Paiva, D. *A Survey of Applied Natural Language Generation Systems*. Rapport technique ITRI-98-03, 1998 (*http://www.itri.brighton.ac.uk/techreports*).

[Panaget 1998] Panaget, F. "Le générateur de langage naturel de l'agent dialoguant artimis". *La génération de textes, t.a.l.*, 39(2), pp. 107–126, 1998.

[Paris 1991] Paris, C., Swartout, W. R. et Mann, W. C. Eds. *Natural Language Generation in Artificial Intelligence and Computational Linguistics*. Kluwer Academic Publisher, 1991.

[Polguère 1998] Polguère, A. "Pour un modèle stratifié de la lexicalisation en génération de textes". *La génération de textes, t.a.l.*, 39(2), pp.57–76, 1998.

[Power 1998] Power, R. J. D. et Scott, D. R. "WYSIWYM: Knowledge editing with natural language feedback". In *Proceedings of the 9th International Workshop on Natural Language Generation (INLG'98)*, pp.308–310, Niagara-on-the-Lake, Ontario, 1998.

[Rambow 1992] Rambow, O. et Korelsky, T. "Applied text generation". In *Proceedings of the 3rd Conference on Applied NLP (ANLP'92)*, pp.40–47, Trento, Italy, 1992.

[Reape 1999] Reape, M. et Mellish, C. Just what *is* aggregation anyway? In *Proceedings of the 7th European Workshop on Natural Language Generation (ENLG'99)*, pp. 20–29, Toulouse, France, 1999.

[Reiter 1994] Reiter, E. " Has a consensus NL generation architecture appeared, and is it psycholinguistically plausible? " In *Proceedings of the 7th International Workshop on Natural Language Generation (INLG'94)*, pp. 163–170, Kennebunkport, Maine, 1994.

[Reiter 1995] Reiter, E. NLG *vs.* templates. In *Proceedings of the 5th European Workshop on Natural Language Generation (ENLG'95)*, pp. 95–105, Leiden, the Nederlands, 1995.

[Rubinoff 1992] Rubinoff, R. *Negotiation, Feedback and Perspective within Natural Language Generation*. PhD thesis, University of Pennsylvania, 1992.

[Saggion 1999] Saggion, H. " Linguistic knowledge in automatic abstracting ". In *Proceedings of ACL'99*, University of Maryland, USA, 1999.

[Sowa 1984] Sowa, J. F. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, Reading, Massachusetts, 1984.

[Teich 1994] Teich E., et Bateman, J. A. " Towards the application of text generation in an integrated publication system ". In *Proceedings of the 7th International Workshop on Natural Language Generation (INLG'94)*, Kennebunkport, Maine, 1994.

[Zock 1988] Zock, M. et Sabah, G., Eds. *Advances in Natural Language Generation: an*

Interdisciplinary Perspective. Pinter Publishers, London, 1988.

[Zock 1992] Zock, M. et Sabah, G., "La génération automatique de textes trente ans déjà, ou presque". *Langages*, 106, pp.33–35, 1992.

[Zock 1994] Zock, M. " Language in action, or learning a language by watching it work ". In 7^{th} *Twente Workshop on Language Technology: Computer-Assisted Language Learning*, Twente, 1994.