Étude et modélisation de la syntaxe des langues à l'aide de l'ordinateur

Analyse syntaxique automatique non combinatoire

Synthèse et Résultats

.

Dossier de synthèse des activités de recherches en vue

de l'obtention du diplôme d'Habilitation à Diriger des Recherches

présenté le 29 septembre 1999

par

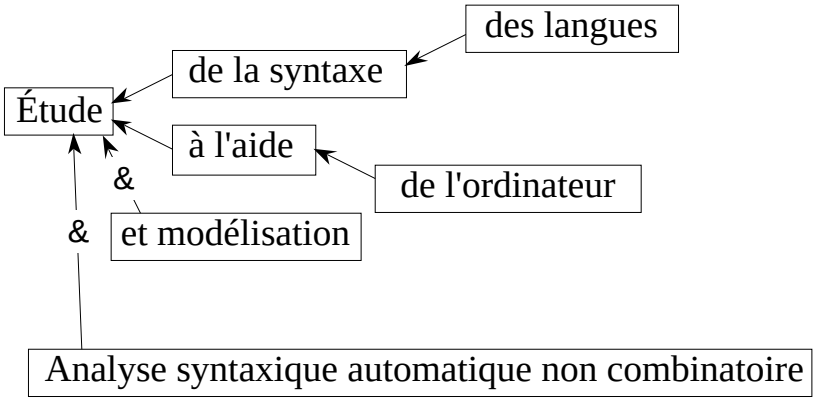
Jacques Vergne

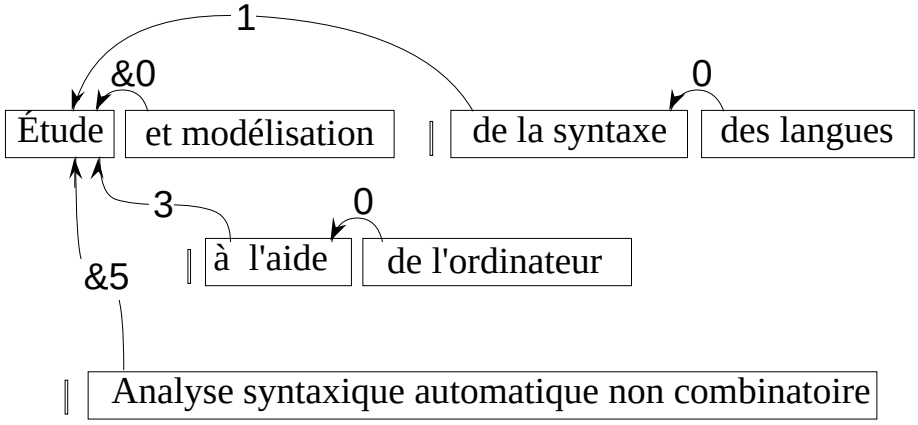
Composition du jury :

Rapporteurs :	Patrice Enjalbert, professeur d'université	Université de Caen
	Violaine Prince, professeur d'université	Université de Paris 8
	Jean Véronis, professeur d'université	Université d'Aix en Provence
Examinateurs :	Daniel Kayser, professeur d'université	Université de Paris 13
	Bernard Victorri, directeur de recherche CNRS	LTM - ENS Ulm
	Pierre Zweigenbaum, chercheur ingénieur	SIM - AP HP

Groupe de **Re**cherches en **I**nformatique, **I**mage, **I**nstrumentation de **C**aen

CNRS - UPRESA 6072


des langues
de la syntaxe
Étude
à l'aide
de l'ordinateur
&
et modélisation
&
Analyse syntaxique automatique non combinatoire


1
&0
0
Étude
et modélisation
de la syntaxe
des langues
3
0
à l'aide
de l'ordinateur
&5
Analyse syntaxique automatique non combinatoire

Résumé introductif

Nous présentons dans ce dossier une synthèse des travaux que nous avons réalisés en syntaxe (linguistique informatique) et en analyse syntaxique automatique (informatique linguistique).

Après les axes de recherches (1.), nous présentons d'abord l'axe "linguistique informatique", constitué principalement par une caractérisation des processus de transformation entre arbre de dépendance et ordre linéaire (2.).

Le deuxième axe, celui de l'"informatique linguistique", montre la progression vers des analyseurs syntaxiques non combinatoires, d'abord analyseurs de phrases, puis analyseurs en flux (3.).

Les actions de validation et de valorisation des recherches sont ensuite présentées (4.), dont la direction scientifique de deux projets industriels de recherche appliquée et de transfert de technologie.

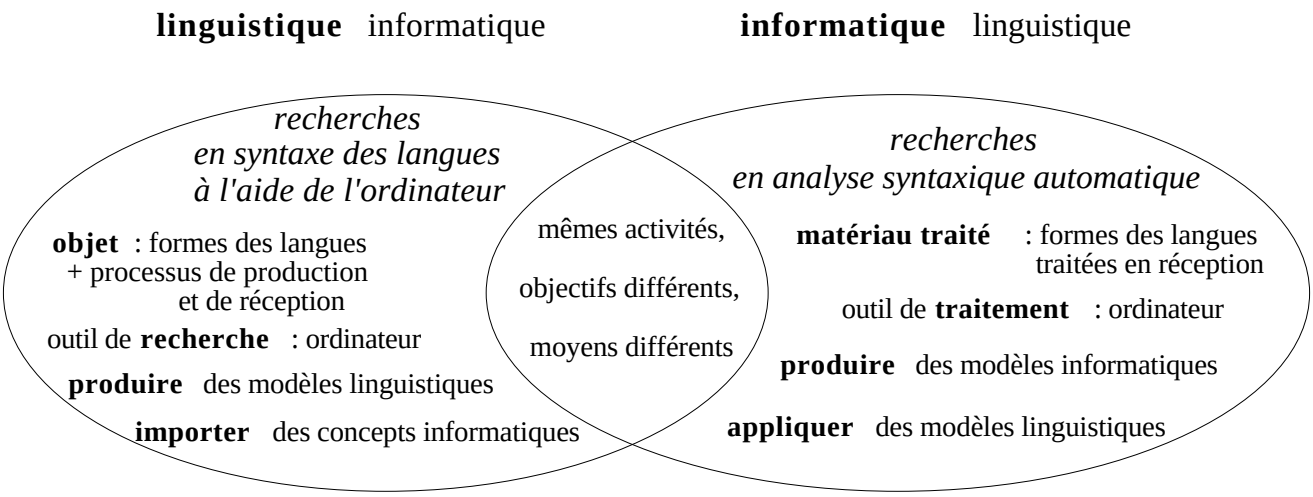
Le chapitre (5.) est consacré à deux regards sur le domaine : les liens forts entre linguistique et informatique, puis la méthode de recherche en syntaxe, fondée sur une confrontation des corpus et de l'ordinateur.

En (6.), nous tentons une synthèse des concepts principaux. Puis viennent une présentation de la prospective de nos recherches (7.) et enfin les publications personnelles, celles des thésards encadrés, et celles qui ont accompagné nos recherches (8.).

Table des matières

1. Axes de recherche

Mes recherches se développent simultanément sur les deux axes conjoints de la **linguistique** informatique et de l'**informatique** linguistique. J'entends par *linguistiqueinformatique* un domaine de recherches en linguistique où l'informatique joue un rôle d'instrument de recherche, à distinguer de l'*informatiquelinguistique*, domaine de recherches en informatique où un matériau linguistique est l'objet traité (voir aussi en 5.1.2.) :



• linguistique informatique **: étude et modélisation de la syntaxe des langues à l'aide de l'ordinateur**

La syntaxe des langues consiste en l'étude des formes (phrases et textes) indépendamment du sens. Dans cet aspect de mes recherches, l'ordinateur est un outil de recherche en syntaxe des langues permettant une méthode expérimentale; il sert à observer le matériau, à modéliser ses propriétés distributionnelles (L. Bloomfield, Z. Harris), à valider des concepts de la linguistique structurale, et à confronter ces concepts avec les corpus (méthode inductive et hypothético-déductive).

Le travail sur corpus situe ces recherches dans le cadre de l'énonciation (la réalité des énoncés attestés), et non pas dans l'idéal de la langue;

L'originalité de l'approche est de chercher d'abord à expliciter les **processus** de production et de réception : ceci revient à expliciter des opérateurs en nombre fini plutôt que des opérandes en nombre infini (expliciter exhaustivement les structures correspond à la démarche classique). Ces processus contraignent les structures, principalement du fait qu'une phrase est un objet à une dimension, et que l'effort de mémoire est minimisé en plaçant les segments reliés les plus proches possible.

Des liens entre forme écrite et forme orale ont été caractérisés comme étant deux instances du même objet; les segments de l'écrit ont leurs correspondants à l'oral : le syntagme simple (non récursif) correspond au groupe accentuel, et l'arbre de dépendance linéarisé implique les groupes prosodiques (la prosodie permet à l'auditeur de calculer les dépendances); ces travaux se concrétisent dans le cadre d'un projet industriel de synthèse vocale.

• **informatique** linguistique **: analyse syntaxique automatique de complexité pratique linéaire sur corpus et production de logiciels prototypes**

L'analyse syntaxique automatique consiste ici à prendre un flux textuel en entrée, et à produire en sortie un texte segmenté (en "mots", syntagmes, propositions et phrases), dont les segments sont catégorisés et reliés. Les analyseurs actuels sont conçus sur le modèle de la compilation et font l'hypothèse de l'exhaustivité des ressources, ce qui revint à considérer une langue comme un langage formel exhaustivement défini; ils cherchent à affecter à une phrase entrante une des structures stockées sous forme d'une grammaire formelle, par un processus combinatoire, de complexité en temps au mieux en $O(n^2)$, n étant le nombre de mots de la phrase, ce qui rend quasi-impossible leur utilisation industrielle au dessus de 20 mots par phrase.

J'ai caractérisé les causes des dysfonctionnements de ces analyseurs et j'ai mis au point un algorithme totalement original, de complexité linéaire, qui fonctionne sur du texte tout venant, et ne fait aucune hypothèse sur les structures globales des phrases. Cet algorithme met en relation des segments par attentes et par l'intermédiaire de mémoires spécialisées par type de relation, et est généralisé pour tout type de relation (dépendances, coordinations, antécédence, coréférences anaphoriques, ...) entre tout type de segment; cet algorithme permet une mise en relation quelles que soient la distance entre les segments reliés et la structure des segments qui les séparent.

L'approche est multilingue : les concepts on été validés sur des corpus de langue variées (cf. Hervé Déjean 1998) et des moteurs indépendants de la langue traitée utilisent des ressources monolingues externes. La méthode d'analyse a été généralisée et abstraite par Emmanuel Giguet (1998), d'où un processus abstrait et générique d'analyse, paramétrable selon l'unité linguistique construite, la tâche réalisée et la langue traitée, ce qui permet de produire du logiciel efficace et léger, car fondé sur ce processus générique et sur les propriétés du matériau linguistique; l'analyse utilise le repérage de discontinuité, de contraste, de frontières, de différences. Le processus abstrait et générique s'instancie dans un moteur générique analogue à un ensemble de systèmes experts dans lesquels les faits sont constitués par le flux entrant, flux sur lequel s'appliquent des règles conditions => actions, à tous niveaux des unités linguistiques.

Les processus sont capables de manipuler des structures partiellement définies, d'où des processus de calcul à partir de données (tels une résolution d'équation) et non pas des processus faisant des choix parmi des attributs ou des structures exhaustivement énumérés et explicités : la machine est considérée comme une machine à calculer et non pas comme une machine à stocker et à choisir dans le stock (processus aptes à "connaître" plutôt que "reconnaître"). Les structures sont calculées en utilisant : (1) des ressources internes (caractéristiques formelles et positionnelles des unités composant la structure à créer), (2) des ressources externes (position et rôle de la structure dans l'unité linguistique de niveau supérieur qui la contient).

2. Linguistique informatique :

Un modèle pour les deux processus de transformation entre arbre de dépendance et ordre linéaire

2.0. Introduction

Dans cette partie (synthèse de notre article des Cahiers de Grammaire n°23 - décembre 1998), nous présentons les deux processus qui permettent à une personne de coder un arbre en une phrase, et à une autre personne de décoder cette phrase en reconstruisant l'arbre, et nous y associons les deux formes structurelles que peut prendre une phrase : son ordre linéaire et son arbre de dépendance. Nous faisons l'hypothèse que cette présentation simultanée, cette appréhension des structures sous les contraintes des processus, permettent une meilleure théorisation de l'ensemble processus - structures.

Dans notre démarche, nous pensons qu'il est plus intéressant et plus faisable d'expliciter les *processus* qui produisent ou interprètent les structures, que de poursuivre sans fin l'énumération exhaustive des *structures*, y compris exprimées sous la forme condensée des grammaires formelles : ceci consiste à caractériser des opérateurs en nombre fini (les processus), au lieu de tenter d'énumérer des opérandes en nombre infini (les structures). En outre, cette démarche est favorisée par le fait que l'ordinateur, notre principal outil de recherche, est plus un outil d'explicitation et d'exécution de processus (les programmes, les algorithmes), que de reconnaissance de structures supposées connues au moment de la conception des algorithmes. Ceci est une intrusion explicite de la chronologie, précédemment mise à l'écart quoique présente au cœur même de l'outil de recherche, l'ordinateur. L'évolution proposée consiste donc à expliciter des processus qui observent, extraient, produisent les structures des phrases analysées, sans attendu préalable sur ces structures. En généralisant, on peut dire qu'un objet statique devient plus intelligible si l'on inclut dans l'étude les processus qui l'ont produit et les processus qui l'exploitent. Bien entendu, les deux processus en jeu ici sont ceux de la production et de la réception dans la situation de communication de deux êtres humains. Ajoutons qu'une telle étude de l'objet, qui inclut les processus *ante* et *post*, fait émerger dans l'objet des traces des processus; plus précisément on trouve dans l'objet observé des marques de *contraintes* qui se sont appliquées dans le processus de production, et des marques de contraintes qui s'appliqueront dans le processus de réception, pour le rendre possible et le faciliter.

Nous tentons ainsi de replacer la phrase linéaire entre le processus qui l'a produite et celui qui va l'interpréter, en étudiant comme un tout la phrase linéaire (objet observable) avec ces deux processus (hypothèses).

Dans la section 2.1, nous présentons les deux processus successifs : production puis réception de la phrase (les processus sur la phrase), ou bien linéarisation puis reconstruction de l'arbre (ces mêmes processus, mais sur l'arbre). La section 2.2 concerne les structures : présentation des segments reliés (les syntagmes réduits), présentation de l'arbre de dépendance à transmettre, et de la phrase linéaire

transmise, c'est-à-dire l'arbre de dépendance linéarisé. La section 2.3 présente et détaille les processus **:** le processus de production (l'opération de linéarisation optimisée de l'arbre) et le processus de réception (la reconstruction de l'arbre par mise en relation des syntagmes). Dans les deux processus, les contraintes mémorielles jouent un rôle central.

2.1. Les deux processus successifs : linéarisation puis reconstruction de l'arbre

Lucien Tesnière définit les deux "ordres" structural et linéaire de la manière suivante dans (Tesnière 59), page 16, § 1 :

1.- L'ordre structural des mots est celui selon lequel s'établissent les connexions.

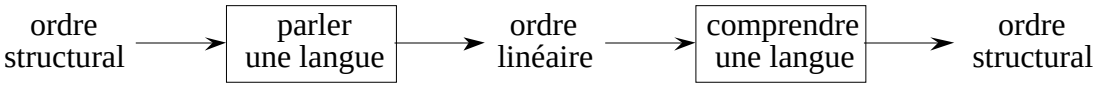
et page 18, § 8 :

8.- Nous appellerons ordre linéaire celui d'après lequel les mots viennent se ranger sur la chaîne parlée. L'ordre linéaire est, comme la chaîne parlée, à une dimension.

Ensuite, page 19, § 4, il présente ainsi les processus :

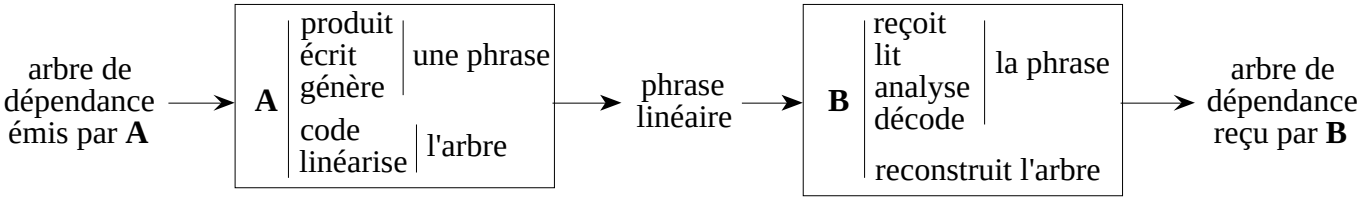
[...] nous pouvons dire que [...] ***parler*** *une langue, c'est en transformer l'ordre structural en ordre linéaire, et inversement que* ***comprendre*** *une langue, c'est en transformer l'ordre linéaire en ordre structural.*

Figure 1 : Les deux processus selon Tesnière, présentés comme successifs



Voici maintenant comment nous concevons les deux processus, placés en situation de transmission d'information : une personne **A** transmet un arbre de dépendance à une personne **B**, en le codant temporairement sous forme de phrase linéaire. Les processus sont désignés en tant qu'actions exécutées par A puis par B, et s'appliquant soit à l'arbre soit à la phrase :

Figure 2 : Les deux processus successifs, placés en situation de transmission d'un arbre de dépendance



Les deux processus sont successifs, réciproques du point de vue fonctionnel, mais différents du point de vue organique, et exécutés par deux personnes différentes, comme nous allons le voir ci-dessous. L'arbre de dépendance est une représentation abstraite hypothétique, alors que la phrase linéaire est le matériau observable.

Au cours de ces deux processus successifs, l'arbre de dépendance est ainsi l'objet, l'information à transmettre entre les deux personnes qui communiquent. Cet arbre de dépendance est temporairement linéarisé (et codé, compressé) dans la phrase linéaire.

Ce sont évidemment des êtres humains qui exécutent ces processus, mais il est possible de les modéliser sur ordinateur, et ceci avec deux objectifs possibles : soit un objectif opératoire de génération ou d'analyse de phrase dans le cadre d'une application informatique plus large (traduction automatique, indexation automatique, correction orthographique), soit un objectif de recherche fondamentale sur la

syntaxe, qui est pour nous l'objectif prioritaire. Cette modélisation est une réduction, une schématisation, une analogie des processus, à usage exploratoire et expérimental.

Les processus et les structures sont façonnés par de nombreuses contraintes :

- contrainte des propriétés géométriques des structures, surtout du fait que la phase linéaire est unidimensionnelle;
- contrainte du fait que l'information contenue dans l'arbre n'est pas perdue au cours de la linéarisation : cette information est codée différemment dans la phrase linéaire;
- contrainte de la chronologie des processus et des déductions;
- contrainte de la minimisation de l'effort de mémoire (voir ci-dessous en 2.3.1), qui implique la minimisation des distances entre nœuds dans l'ordre linéaire [1].

Les deux structures sont difficilement théorisables statiquement, en dehors des deux processus, comme le montrent les tentatives des linguistes et des chercheurs en traitement automatique des langues (TAL). Les linguistes ont surtout tenté de modéliser arbre et phrase linéaire de manière statique : la génération chez Chomsky reste le processus de déduction de la démarche hypothético-déductive, (Mel'cuk 88) dit page 129 sa difficulté à définir sa "*syntactic dependency*", et la connexion chez Tesnière est définie comme un processus de perception et de calcul (voir ci-dessous en 2.3.2.2) mais le concept d'arbre de dépendance (le stemma) est resté statique, et la correspondance entre ordre linéaire et ordre structural, présentée sous la forme des deux processus, n'est pas développée, mais seulement esquissée par le terme de "image projetée", uniquement dans le paragraphe suivant, page 20, § 10 :

[...] syntaxiquement, la vraie phrase, c'est la ***phrase structurale*** *dont la phrase linéaire n'est que l'image projetée tant bien que mal, et avec tous les inconvénients d'aplatissement que comporte cette projection sur la chaîne parlée.*

Quant à l'analyse syntaxique automatique classique, c'est un processus combinatoire d'essais de tous les choix possibles jusqu'à la reconnaissance de structures explicitement attendues, processus tout à fait différent du processus de reconstruction de l'arbre présenté ici (voir ci-dessous en 2.3.2).

Notre démarche est de faire des hypothèses sur l'ensemble des deux structures et des deux processus, des hypothèses explicites sur les processus, qui conduisent à définir implicitement les structures à partir des processus. Dans le travail présenté ici, nous proposons :

- pour le processus de linéarisation de l'arbre (en 2.3.1), des hypothèses sur le parcours d'arbre issues de l'informatique théorique, confrontées à une étude sur corpus ;

- pour le processus de reconstruction de l'arbre (en 2.3.2), des hypothèses sur un processus de mise en relation fondé sur la mémoire, modélisées et généralisées dans le cadre d'un analyseur syntaxique, puis confrontées à l'analyse automatique d'un corpus.

2.2. Structures

Cette section présente d'abord (en 2.2.1) le segment (le syntagme non récursif) qui est le constituant de base des deux structures : l'arbre de dépendance à transmettre (2.2.2), et la phrase linéaire transmise, c'est-à-dire l'arbre de dépendance linéarisé (2.2.3).

[1] Cette hypothèse du moindre effort est déjà ancienne : on la trouve déjà chez (Zipf 1949), et plus récemment chez (Grunig 1993), page 18, au sujet du moindre effort de mémoire : "*Il apparaîtra que certains types d'encodages (et décodages associés) sont, du seul point de vue des charges mémorielles que je retiens ici (...), clairement plus économiques que d'autres*".

2.2.1 Les segments reliés = les Syntagmes Non Récursifs (SNR)

Observons comment, dans Le Bourgeois Gentilhomme (figure 3), Molière fait permuter des groupes de mots dans la phrase, mais laisse inchangé l'ordre des mots dans chaque groupe :

Figure 3 : Définition implicite d'un segment permutable par Molière

Belle marquise	, vos beaux yeux	me font	mourir	d'amour	.
D'amour	mourir	me font	, belle marquise	, vos beaux yeux	.
Vos beaux yeux	d'amour	me font	, belle marquise	, mourir	.

Le segment de phrase choisi pour être le segment relié par les relations de dépendance est non pas le mot, dont la pratique est conventionnelle et instable entre des langues différentes, mais ce groupe de mots que Molière fait permuter : le syntagme non récursif, ou syntagme simple, ou syntagme noyau, ou "core phrase", ou "chunk" dans la littérature en anglais (voir Abney 1996), ou syntagme sans ses syntagmes compléments, par opposition au syntagme récursif des grammaires génératives, qui inclut ses syntagmes compléments.

Dans la suite de ce chapitre, nous appellerons ce segment SNR, comme Syntagme Non Récursif. Il sera délimité par un rectangle dans toutes les figures, ce qui donnera au lecteur des exemples par la pratique. Par définition (non exhaustive), un SNR est constitué d'un élément central, le plus souvent un nom (ou un pronom tonique) ou un verbe (conjugué, infinitif, participe présent ou passé), entouré éventuellement de ses éléments périphériques :

- dans le SNR nominal : conjonction de coordination et/ou de subordination, préposition, déterminant, adjectif épithète antéposé ou postposé, adverbe antéposé à l'adjectif épithète; exemples : de *avec appétit* à *mais avec une très belle raquette blanche*;
- dans le SNR verbal : conjonction de coordination et/ou de subordination, préposition, tous les pronoms atones (sujet, objet et autres) antéposés ou postposés, négations, auxiliaire, adverbe le plus souvent postposé, adjectif attribut avec la copule être, adverbe antéposé à l'adjectif attribut; exemples : de *ne voit* à *qu'il ne le leur a sûrement pas donné hier*.

L'élision est interne au SNR (le mot élidé et le mot suivant appartiennent au même SNR). Une ponctuation ne peut couper un SNR. Le SNR nominal est marqué par un genre et un nombre homogènes sur tous ses composants variables en flexion. Ce segment montre sa stabilité sur des langues variées comme on peut le voir dans les travaux de Hervé Déjean (cf. Déjean 1998).

En se référant aux définitions possibles du "groupe verbal" par (Le Goffic 1993), page 29, le SNR ne correspond ni à sa "définition large" (le verbe avec tous ses compléments = le prédicat complet = le syntagme verbal récursif), ni à sa "définition étroite" (le verbe avec son auxiliaire éventuel, en excluant tout pronom atone), mais se situe entre les deux [2].

Le SNR permet de définir une hiérarchie de segments à quatre niveaux : mots, SNR, propositions, phrases, hiérarchie où le segment d'un niveau est constitué de segments du niveau inférieur : un tout est d'un type différent du type de ses parties, contrairement au syntagme récursif, constitué de mots et de syntagmes récursifs. Les mots dans un SNR, les SNR dans une proposition, et les propositions dans une phrase ont des comportements très différents : les mots d'un SNR sont dans un ordre très contraint autour d'un nom ou d'un verbe, mais les SNR dans une proposition sont dans un ordre soumis à des contraintes plus relâchées.

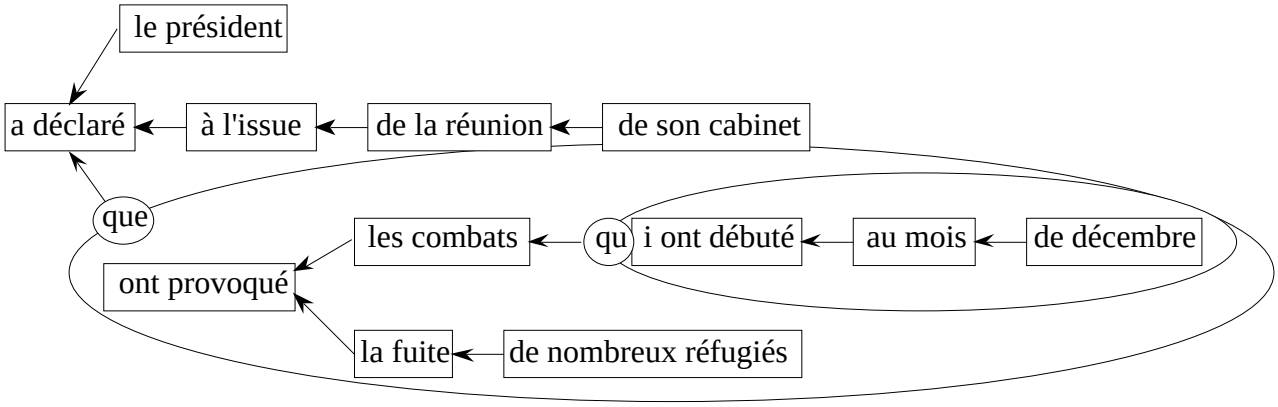
[2] Remarque : notre définition "médiane" du SR correspond à une définition possible du groupe accentuel (qui contient un seul accent primaire, et éventuellement des accents secondaires); les phénomènes d'élision, liaison et enchaînement sont internes au groupe accentuel ainsi défini et permettent des tests en cas d'hésitation sur ses limites.

Notons que, dans notre démarche, constituance [3] et dépendance ne sont ni déclarées équivalentes (Gaifman 1965), ni opposées, mais utilisées ensemble à deux niveaux différents : constituance à l'intérieur des SNR, dépendance entre SNR.

2.2.2 L'arbre de dépendance à transmettre

Dans l'arbre de dépendance à transmettre, théoriquement identique à l'arbre reçu, les segments reliés (i.e. les nœuds de l'arbre) sont des SNR (et non pas des mots comme pour Tesnière), et ils sont reliés par des relations de dépendance. Le nœud-racine conventionnel est le SNR verbal de la proposition principale.

Figure 4 : L'arbre de dépendance à transmettre



En voici un exemple, dans la figure 4, où l'arbre est dessiné horizontalement, pour maintenir l'écriture horizontale des SNR et des suites de SNR dépendants, et dans lequel le SNR-racine est conventionnellement placé à gauche. Remarquons que nous avons toute liberté de dessin de l'arbre, car nous nous libérons de la contrainte de projectivité (la projectivité est la propriété d'être projetable pour un arbre de dépendance) en concevant la linéarisation différemment (par parcours de l'arbre, voir ci-dessous en 2.3.1.1); cet abandon de la contrainte de projectivité est en quelque sorte un retour au stemma de Tesnière, qui représente l'ordre structural dissocié de l'ordre linéaire.

Nous proposons une représentation particulière de la proposition subordonnée : le régissant de sa hiérarchie interne (appelé souvent sa "tête") est son SNR verbal conjugué; mais, vue de son extérieur, la proposition subordonnée constitue un tout (entouré d'un ovale), relié à son régissant par l'intermédiaire de la conjonction de subordination (entourée d'un cercle) qui sera linéarisée au début de la subordonnée. On dissocie ainsi deux questions habituellement liées : d'une part la subordonnée inclut le régissant interne de sa hiérarchie interne, et d'autre part c'est la subordonnée entière (et non pas un de ses éléments) qui dépend de son régissant externe (comme le SNR entier dépend de son régissant externe)[4].

2.2.3 La phrase linéaire transmise = l'arbre de dépendance linéarisé

La phrase linéaire transmise est le résultat du processus de linéarisation de l'arbre de dépendance (décrit ci-dessous en 3.1). Les segments et les dépendances sont les mêmes que dans l'arbre à transmettre.

---

[3] NB : pour ce néologisme, on trouve les deux orthographes *constituance* et *constituence* dans la littérature. L'orthographe *constituance* est plus fréquente. Nous l'avons préférée en tant que dérivée de *constituant,* comme *dépendance* est dérivé de *dépendant.*

[4] Voir une représentation analogue : les "arbres à bulles" de (Kahane 97). Voir aussi la translation du second degré de Tesnière : "La translation I>>O" (qui translate un verbe en substantif) dans (Tesnière 1959), page 546, chapitre 241, translation dont le translatif est la conjonction de subordination *que.*

Pour pouvoir mesurer les longueurs des dépendances, et mesurer et comparer des longueurs de constituants, nous avons défini une **métrique** sur la phrase linéaire de la manière suivante :

- l'unité de la métrique est le Syntagme Non Récursif (SNR), délimité par un rectangle dans les figures;

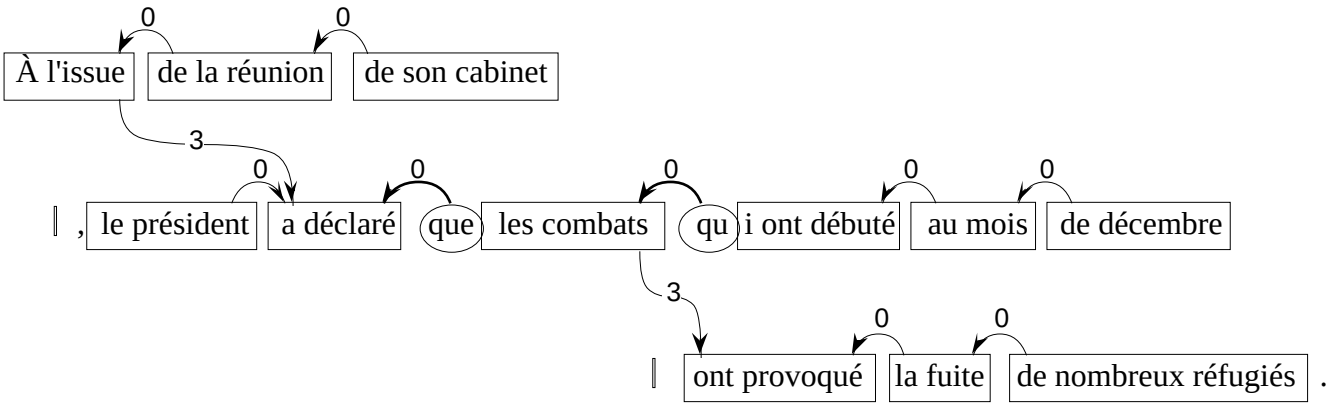
- la longueur de la dépendance entre 2 SNR est le nombre de SNR qui les séparent, ce qui implique que la longueur de la dépendance entre 2 SNR dépendants contigus est nulle;

- la longueur d'un groupe de SNR contigus est le nombre de SNR qu'il contient;

- par définition, la longueur de la dépendance entre une proposition subordonnée et son régissant est le nombre de SNR qui séparent ce régissant du début de la proposition subordonnée (sa conjonction de subordination, ou son pronom relatif); en étant définie ainsi, la distance entre une proposition subordonnée et son régissant reste indépendante de la linéarisation interne de la subordonnée (cette dépendance particulière est marquée par une flèche plus épaisse dans les figures des arbres linéarisés).

En marquant chaque dépendance par une flèche accompagnée de la longueur de la dépendance, on obtient alors l'arbre de dépendance linéarisé de la figure 5.

Figure 5 : Phrase linéaire transmise = arbre de dépendance linéarisé [5]



2.3. Processus

Nous allons présenter dans cette section les deux processus de production et de réception, dans l'ordre chronologique de la transmission de l'arbre de dépendance émis, linéarisé dans la phrase linéaire (2.3.1), puis ensuite reconstruit (2.3.2).

2.3.1 Le processus de production : arbre → phrase linéaire = opération de linéarisation optimisée

Nous allons étudier comment, à la production, la phrase linéaire est produite à partir de l'arbre de dépendance, comment un régissant peut être linéarisé par rapport à ses dépendants, comment les dépendants sont linéarisés, et enfin quels sont les critères de calcul de la linéarisation. Nous terminerons cette section par une étude de la linéarisation sujet - verbe dans les relatives en *que,* étude théorique fondée sur les concepts exposés dans cette sous-section, puis corroborée sur corpus.

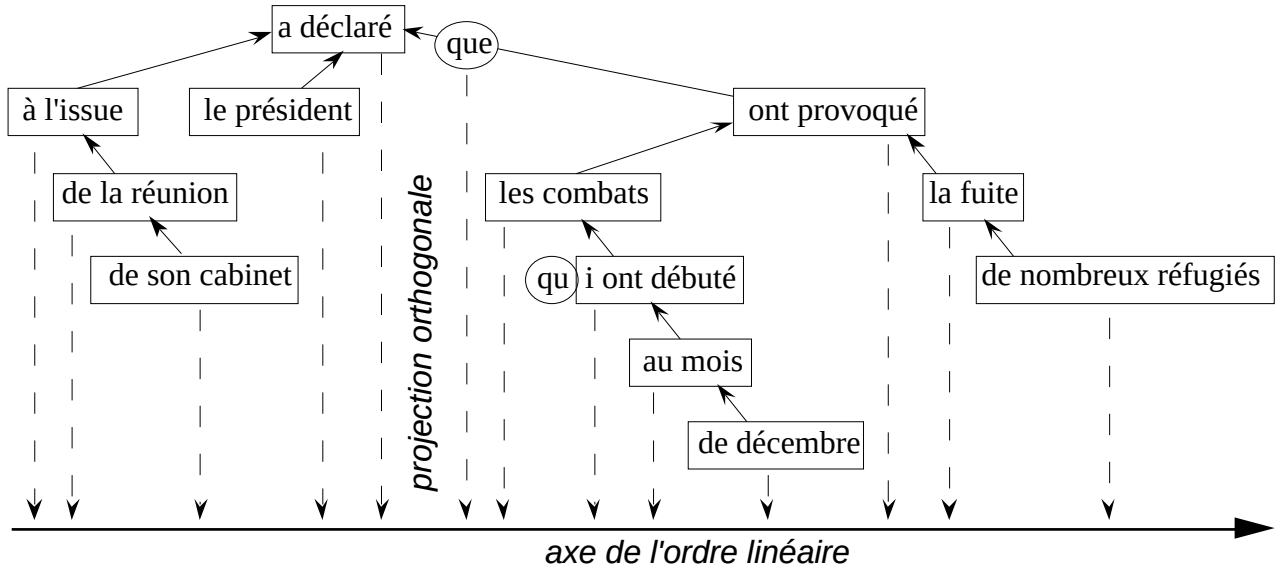
[5] Remarque sur la figure 5 : cette figure est disposée en trois lignes, et chaque ligne est une suite de SR dépendants contigus (reliés par une dépendance de longueur nulle), ce qui constitue une définition possible du groupe prosodique, c'est-à-dire une suite de groupes accentuels dits d'un seul tenant, sans pause; la frontière entre deux groupes prosodiques est placée entre deux SR contigus non directement reliés (frontière marquée par le signe ‖ dans la figure 5, et dans la suite de ce chapitre); ceci établit un lien précis entre la structure de l'arbre de dépendance linéarisé et cette définition du groupe prosodique.

2.3.1.1 Linéariser l'arbre = le parcourir en relevant les nœuds

La linéarisation classique de l'arbre de dépendance consiste en sa projection géométrique orthogonale sur l'axe de l'ordre linéaire, qui a été proposée par (Hays 1964), puis adoptée dans la communauté des Traitements Automatiques des Langues (TAL). Pour rendre projetable l'arbre de la figure 4, il faudrait le dessiner comme dans la figure 6 ci-dessous, laborieusement, car sous deux contraintes difficilement compatibles (et quelquefois incompatibles) : placer un dépendant à la fois au dessous de son régissant et après le syntagme contigu précédent, tout en maintenant la projectivité de l'arbre.

La linéarisation par projection a au moins les trois inconvénients suivants : 1) l'ordre linéaire dépend de la manière dont l'arbre est dessiné; 2) elle a donné naissance au concept de projectivité, qui était un intermédiaire nécessaire pour montrer l'équivalence formelle "*entre la génération par un système de constituance et la génération par un système de dépendance*" possédant un vocabulaire non terminal et reposant sur la projectivité - voir (Portine 1992), page 123, et (Gaifman 1965); et 3) la non-projectivité de phrases extraites de corpus est avérée (par exemple en cas d'incise). L'arbre projetable, contrairement au stemma, a ainsi l'ambition de coder simultanément dans un même schéma à la fois l'ordre structural et l'ordre linéaire (comme l'arbre syntagmatique d'ailleurs).

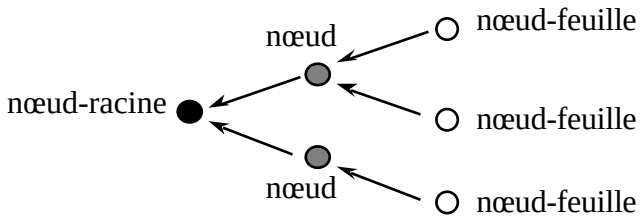
Figure 6 : Arbre de dépendance projetable à la manière de Hays



Nous proposons que l'ordre linéaire soit non pas lisible directement sur l'arbre projetable, mais qu'il soit le résultat d'un **processus** sur la structure de l'arbre à transmettre. Nous abandonnons alors simplement la projection pour le parcours d'arbre, qui ne dépend pas du dessin de l'arbre, mais uniquement de sa structure.

Produire une phrase, c'est placer les segments reliés sur l'axe de la phrase (espace à une dimension), c'est-à-dire énumérer les nœuds de l'arbre de dépendance dans un certain ordre. L'informatique théorique apporte des concepts clairs et adéquats au sujet des arbres et des ordres possibles d'énumération des nœuds d'un arbre : le processus de parcours d'arbre avec relevé des nœuds. La figure 7 rappelle les éléments d'un arbre, et la définition des nœuds et branches de l'arbre de dépendance :

Figure 7 : Nœuds et branches de l'arbre de dépendance



nœud = segment = constituant
= Syntagme Non Récursif (SNR)

branche = relation entre 2 nœuds
= dépendance entre 2 SNR

Parcourir un arbre, c'est passer par tous ses nœuds en suivant un certain chemin, en partant de la racine, et en revenant à la racine; relever les nœuds d'un arbre, c'est parcourir l'arbre en "ramassant" au passage chaque nœud une seule fois, ce qui revient à énumérer les nœuds de l'arbre dans un certain ordre.

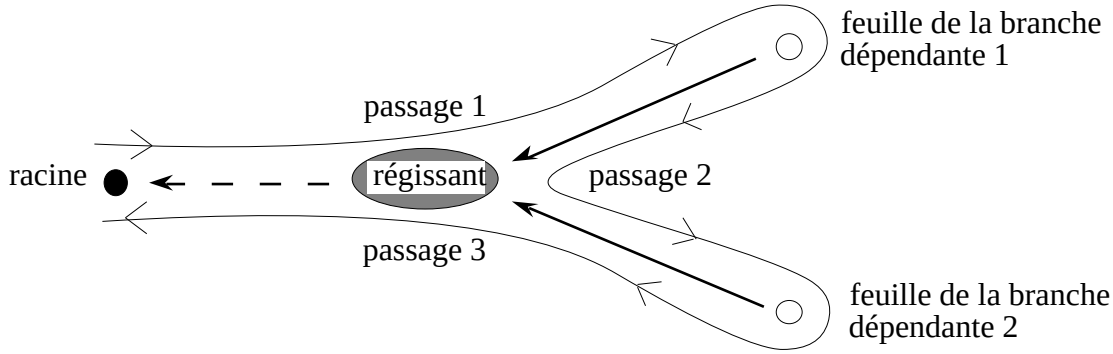
Nous pouvons alors considérer que linéariser un arbre revient à le parcourir en relevant chaque nœud une fois.

On observe que ce relevé est le plus souvent celui qui place les nœuds reliés les plus proches possible (contigus si possible) dans l'ordre linéaire, et nous faisons l'hypothèse que c'est pour **minimiser** les efforts de mémoire à la production et à la réception.

Le parcours d'arbre qui minimise les distances entre nœuds s'appelle le parcours en profondeur d'abord : aller de la racine vers les feuilles, et retour des feuilles vers la racine, en suivant toutes les branches. Pour un nœud à deux branches, une première branche est parcourue en entier avant de parcourir la deuxième branche en entier.

Dans un tel parcours d'arbre, on passe trois fois sur un nœud-régissant qui a deux branches dépendantes (voir figure 8) :

Figure 8 : Parcours d'arbre en profondeur d'abord



aller : racine → nœud-régissant → feuille de la branche dépendante 1,
retour : feuille de la branche dépendante 1 → nœud-régissant,
aller : nœud-régissant → feuille de la branche dépendante 2,
retour : feuille de la branche dépendante 2 → nœud-régissant → racine.

Le type de **relevé** du nœud régissant spécifie à quel passage il est relevé :
- au passage 1, le régissant est relevé **avant** ses dépendants (relevé dit **préfixé**)
- au passage 2, le régissant est relevé **entre** ses dépendants (relevé dit **infixé**)
- au passage 3, le régissant est relevé **après** ses dépendants (relevé dit **postfixé**).

Dans un même parcours d'arbre, chaque nœud a son propre type de relevé. En français, le relevé normal est le relevé préfixé : linéarisation régissant puis dépendants.

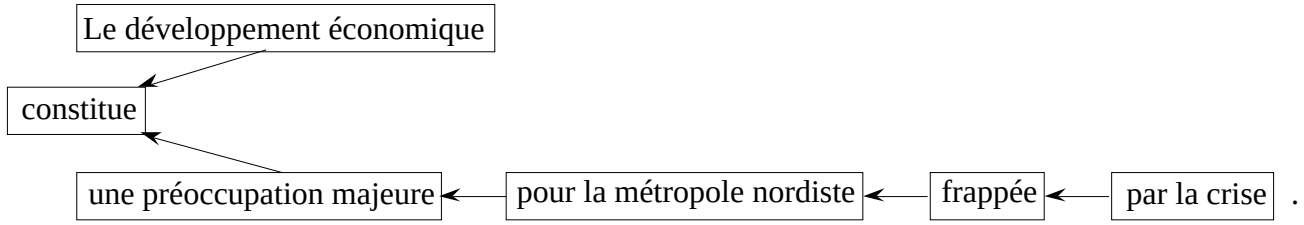
Ces concepts vont nous permettre de catégoriser les différents ordres linéaires possibles entre un nœud régissant et ses nœuds dépendants (voir ci-dessous en 2.3.1.2).

2.3.1.2 Linéarisation d'un nœud régissant par rapport à ses dépendants

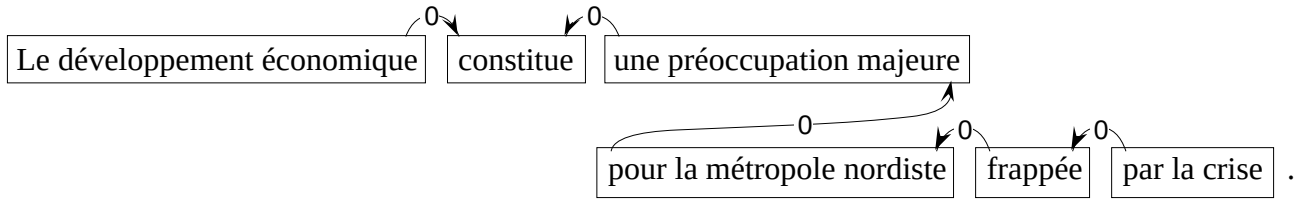
Quand, dans une phrase, un nœud A est relié uniquement à un nœud B, le nœud B peut être placé avant ou après A : A - B ou B - A et les deux nœuds reliés sont contigus; quand un nœud A est relié uniquement à deux nœuds B et C, ces deux nœuds peuvent être placés autour de A : B - A - C ou C - A - B et les nœuds reliés sont toujours contigus deux à deux. Dans la figure 9, voici un exemple de phrase où les nœuds sont reliés à un ou deux autres nœuds, et où toute relation peut ainsi être marquée par une contiguïté :

Figure 9 : Phrase où tous les nœuds reliés sont contigus

arbre de dépendance à transmettre :



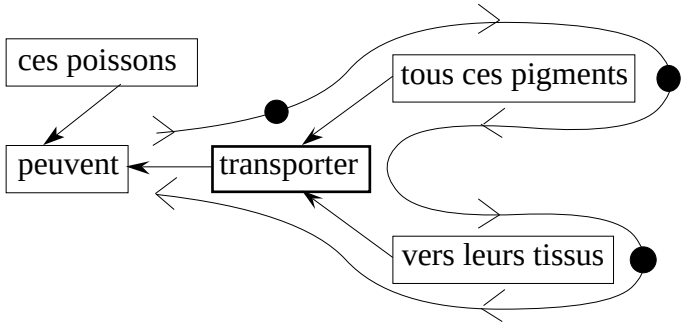
arbre de dépendance linéarisé = phrase linéaire :



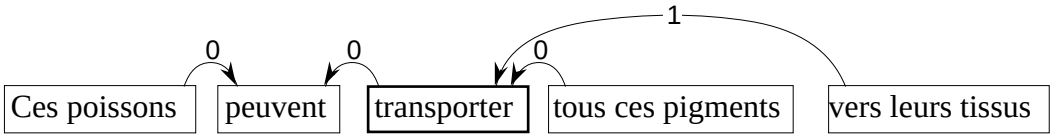
Mais quand un nœud A est relié à trois autres nœuds B, C et D, il n'y a que deux contiguïtés disponibles autour de A, et une des trois relations à A ne peut se faire en contiguïté avec A : c'est une contrainte de l'espace à une dimension, ou bien un des "*inconvénients d'aplatissement*" dont parle Tesnière. En particulier, quand un nœud régissant (qui a lui-même son régissant) a deux nœuds dépendants, ils ne peuvent le suivre tous les deux en contiguïté, alors trois relevés du nœud régissant sont possibles. Dans les exemples des figures 10, 11 et 12), nous allons présenter un même arbre de dépendance, linéarisé de 3 manières différentes :

Figure 10 : Nœud régissant relevé au passage 1, donc **préfixé** avant ses nœuds dépendants

parcours de l'arbre de dépendance à transmettre (● = relevé du nœud) :

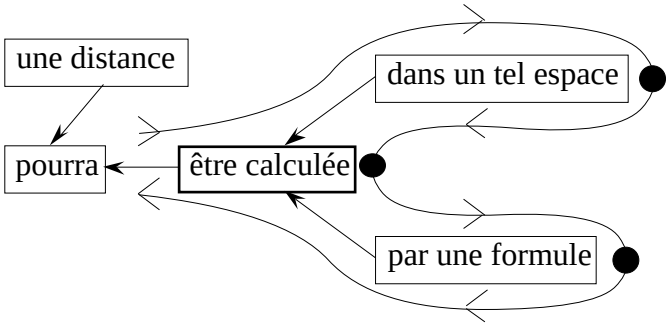


arbre de dépendance linéarisé = phrase linéaire :

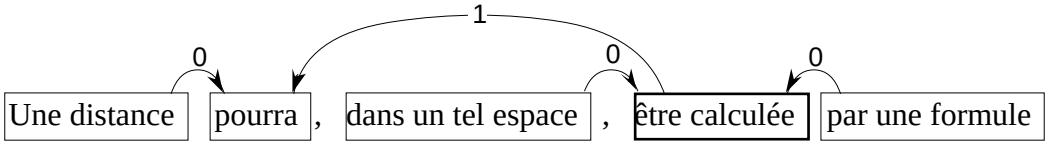


L'actant objet du verbe lui est contigu (à distance nulle), et le complément circonstant du verbe est à distance 1, après l'objet.

Figure 11 : Nœud régissant relevé au passage 2, donc **infixé** entre ses dépendants
parcours de l'arbre de dépendance à transmettre (● = relevé du nœud) :

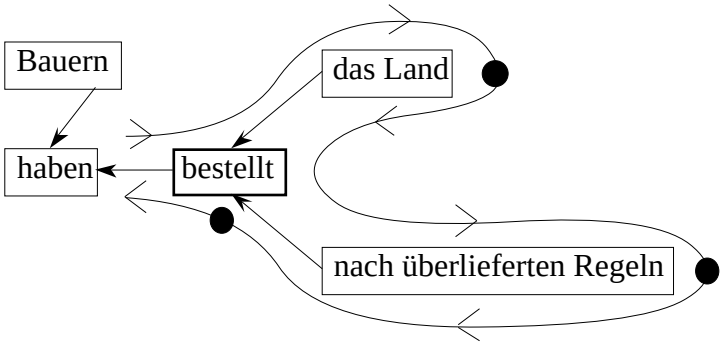


arbre de dépendance linéarisé = phrase linéaire :

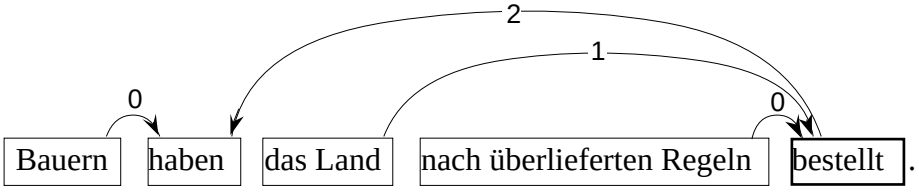


Les deux compléments circonstants de *êtrecalculée* lui sont contigus, et son régissant *pourra* est à distance 1.

Figure 12 : Nœud régissant relevé au passage 3, donc **postfixé** après ses nœuds dépendants
parcours de l'arbre de dépendance à transmettre (● = relevé du nœud) :



arbre de dépendance linéarisé = phrase linéaire :



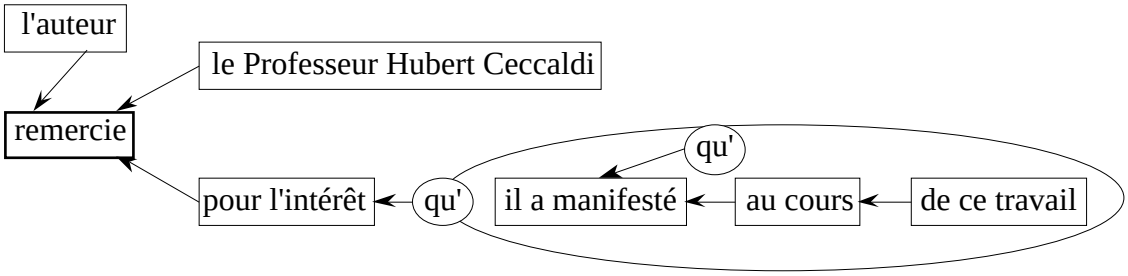
Le cas du participe passé postfixé est classique en allemand : *bestellt* est contigu à son complément circonstant, à distance 1 de son objet, et à distance 2 de son régissant *haben*.

2.3.1.3 Linéarisation des nœuds dépendants

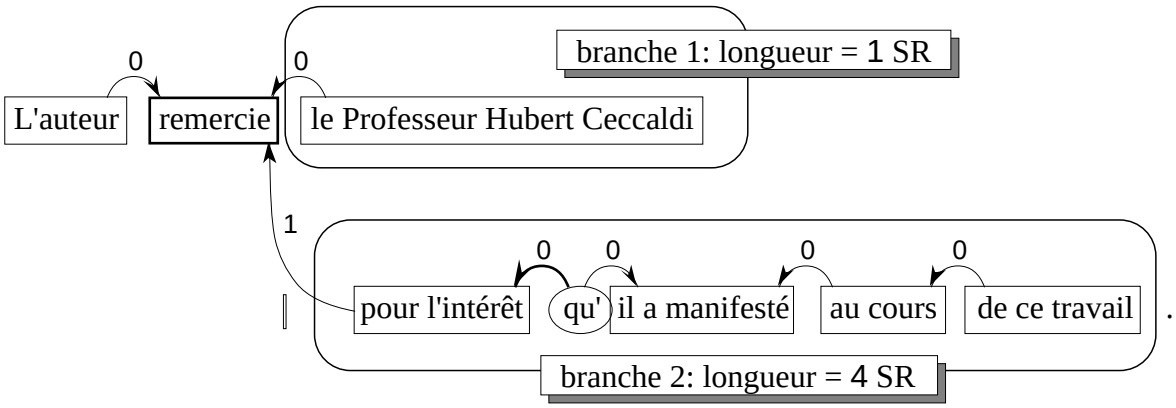
Demandons-nous maintenant quel est l'ordre des nœuds dépendants après le nœud régissant préfixé. Prenons comme exemple le cas où un verbe (nœud régissant préfixé) a un actant objet direct et un complément circonstant indirect (deux nœuds dépendants). Les deux ordres possibles sont soit verbe puis branche objet puis branche complément circonstant, soit verbe puis branche complément circonstant puis branche objet, chaque branche étant parcourue en entier (figures 13 et 14).

Dans l'arbre de dépendance à transmettre de la figure 13, le pronom relatif *que* est présent deux fois : en tant que conjonction de subordination, et en tant que pronom objet.

Figure 13 : Linéarisation : verbe, puis branche objet, puis branche circonstant
arbre de dépendance à transmettre :

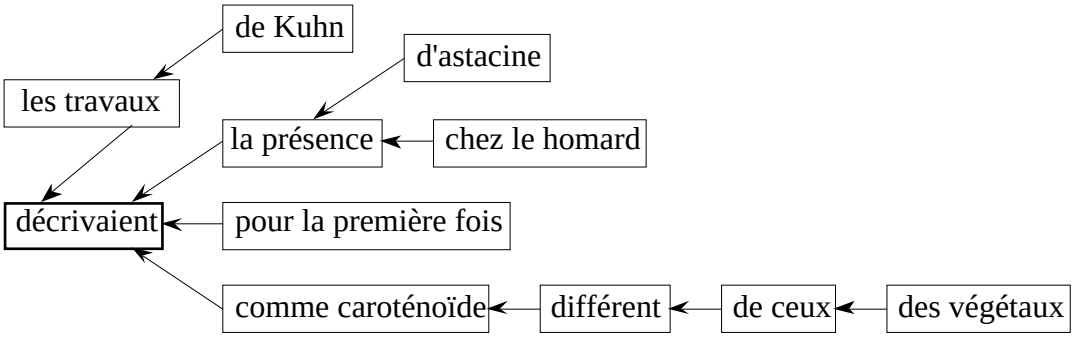


arbre de dépendance linéarisé = phrase linéaire :

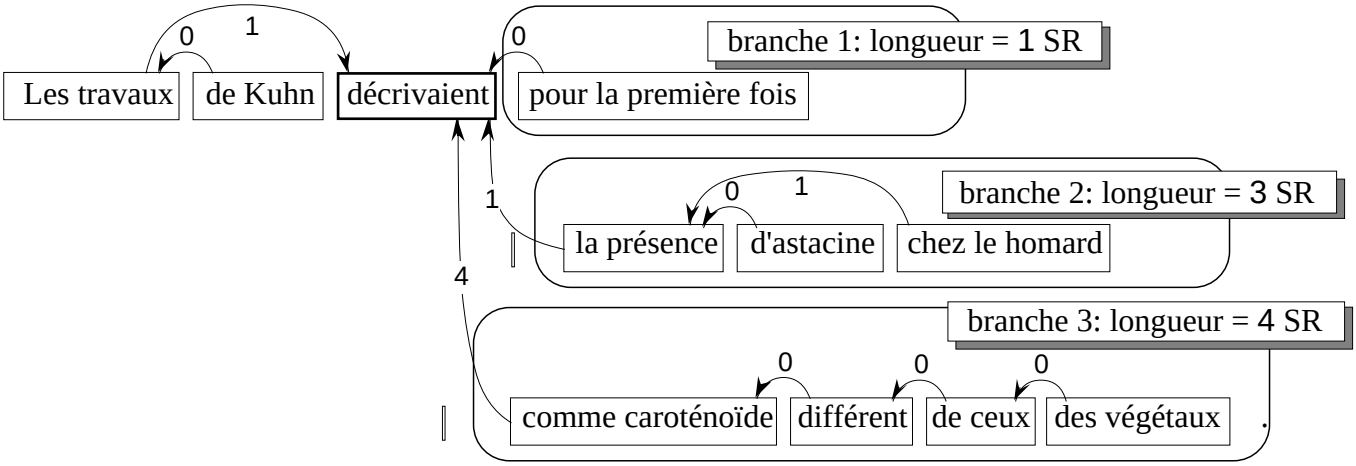


Dans la phrase linéaire de la figure 13, les longueurs des deux branches dépendantes du verbe *remercie* sont mises en évidence : successivement longueurs 1 puis 4.

Figure 14 : Linéarisation : verbe, puis branche circonstant, puis branche objet
arbre de dépendance à transmettre :



arbre de dépendance linéarisé = phrase linéaire :



Dans la phrase linéaire de la figure 14, les longueurs des trois branches dépendantes du verbe *décrivaient* sont mises en évidence : successivement longueurs 1 puis 3 puis 4.

Dans les deux cas, la branche la plus courte est linéarisée la première : des deux ordres linéaires possibles entre deux branches, c'est celui qui minimise la somme des distances entre nœuds reliés. Cette minimisation des distances permet une minimisation de l'effort de mémoire à l'émission (et à la réception) et permettra le calcul des relations à la réception.

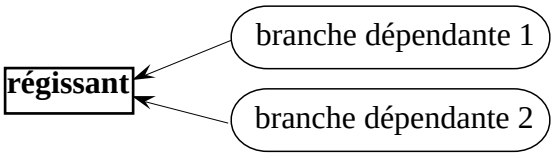
Nous appellerons ce processus "linéarisation optimisée de l'arbre de dépendance".

2.3.1.4 Calcul du choix de la linéarisation optimisée

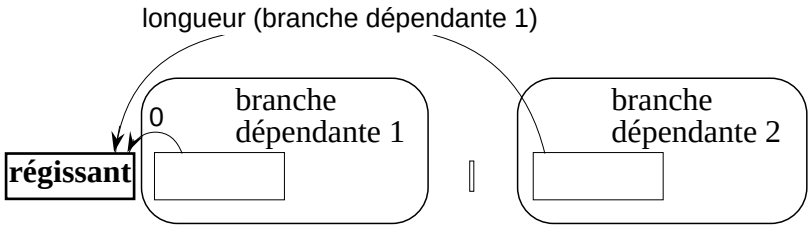
Nous allons maintenant démontrer cette propriété dans le cas normal en français du nœud régissant préfixé, en prenant le cas général abstrait d'un nœud régissant suivi de ses deux nœuds dépendants (un SNR régissant préfixé à ses deux dépendants). Dans la figure 15, la branche 1 est conventionnellement linéarisée la première.

Figure 15 : Linéarisation d'un nœud régissant à deux branches dépendantes

arbre de dépendance à transmettre :



linéarisation : régissant puis dépendants, par relevé préfixé



La longueur de la dépendance du premier nœud de la branche dépendante 1 est 0 car cette branche est produite la première, contiguë à son régissant. La longueur de la dépendance du premier nœud de la branche dépendante 2 est longueur (branche dépendante 1) car cette branche est produite dès que la branche 1 se termine.

L'hypothèse du **moindre effort de mémoire** nous conduit à une définition géométrique du critère d'optimisation de la linéarisation :

la linéarisation optimisée est celle qui, parmi toutes les linéarisations possibles, minimise la somme des longueurs des dépendances.

Pour l'ordre linéaire branche 1 suivie de la branche 2 (cas de la figure 15), la somme des longueurs des dépendances est : longueur (branche dépendante 1)

Pour l'ordre linéaire branche 2 puis branche 1, la somme des longueurs des dépendances est : longueur (branche dépendante 2).

La linéarisation optimisée est donc l'ordre linéaire branche 1 puis branche 2 si :

longueur (branche dépendante **1**) ≤ longueur (branche dépendante **2**)

la linéarisation optimisée place en premier la branche la plus courte (pour un nœud régissant suivi de ses deux nœuds dépendants)

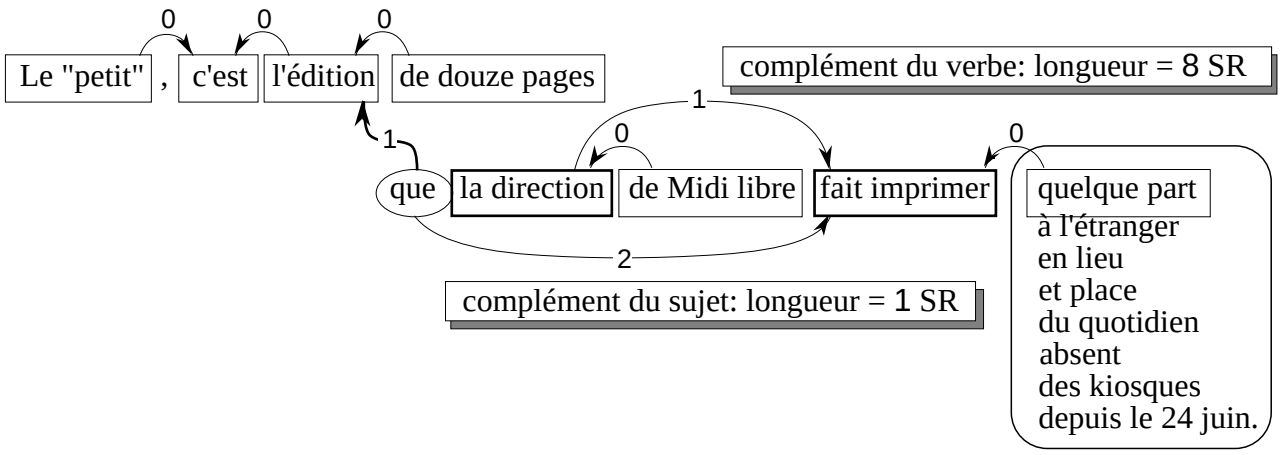
En généralisant à un nombre quelconque de branches, la linéarisation optimisée est donc l'ordre linéaire qui place les branches par ordre croissant de longueur : c'est le cas de la figure 14 où l'on a l'ordre linéaire des trois branches de longueurs 1, 3 et 4.

2.3.1.5 Linéarisation de la proposition relative en *que*

La proposition relative est bien connue pour illustrer la question de l'ordre sujet - verbe ou verbe - sujet. Dans cette sous-section, nous proposons d'abord deux exemples, puis une étude théorique, puis enfin une étude sur corpus.

Voici tout d'abord deux exemples de propositions relatives extraites du journal Le Monde. Dans la figure 16, le sujet de la relative, placé avant le verbe, a un complément de longueur 1, le verbe un complément de longueur 8. Le pronom relatif objet est à une distance 2 du verbe *fait imprimer* dont il sature la valence objet.

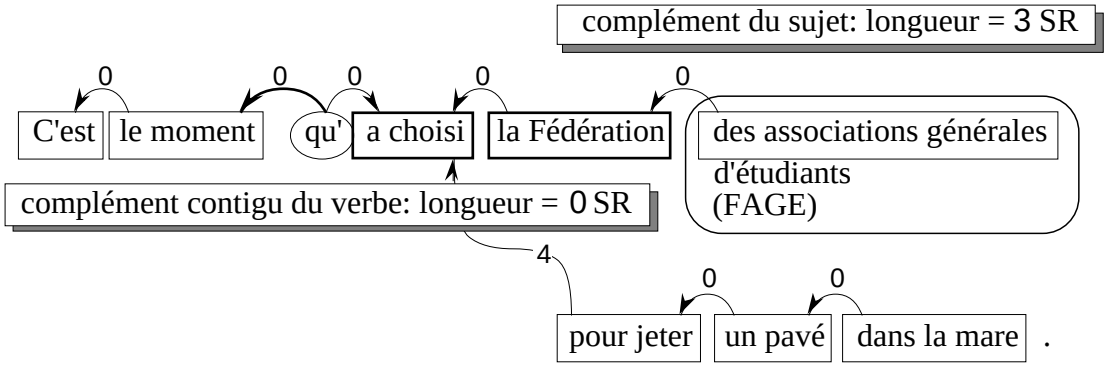
Figure 16 : exemple de linéarisation sujet - verbe



Dans la figure 17, le sujet de la relative, placé après le verbe, a un complément de longueur 3, le verbe n'a aucun complément contigu. Il a aussi des compléments non contigus, rejetés après les

compléments du sujet, et qui ne font pas partie de la structure actancielle du verbe *a choisi* . Le pronom relatif objet est à une distance 0 du verbe *a choisi* dont il sature la valence objet.

Figure 17 : exemple de linéarisation verbe - sujet



Nous allons maintenant faire l'étude théorique de la linéarisation de la proposition relative en *que*, dans le cas où le sujet n'est pas un pronom sujet atone, mais un syntagme (qui peut donc avoir des compléments), et où le pronom relatif sature directement la valence objet du verbe de la relative, comme dans les figures 16 et 17, et non pas la valence objet d'un verbe infinitif ou conjugué dépendant du verbe de la relative, comme dans l'exemple suivant (voir son arbre linéarisé en annexe A.2 de notre article des Cahiers de Grammaire) :

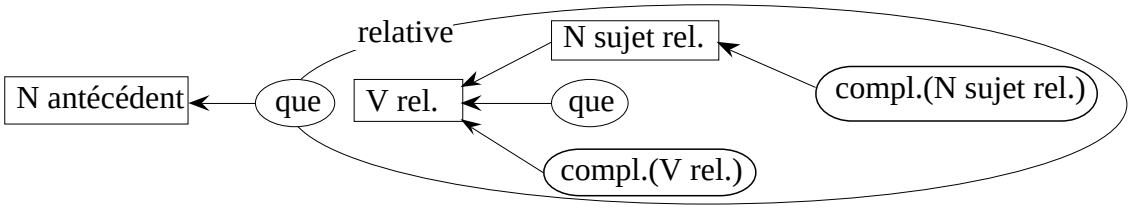
> *Ce document confirme l'importance des pertes **que** l'Etat sera amené à **supporter** sur les actifs détenus par le CDR.*

exemple où ***que*** sature la valence objet de l'infinitif ***supporter*** qui dépend de *sera amené* verbe conjugué de la relative.

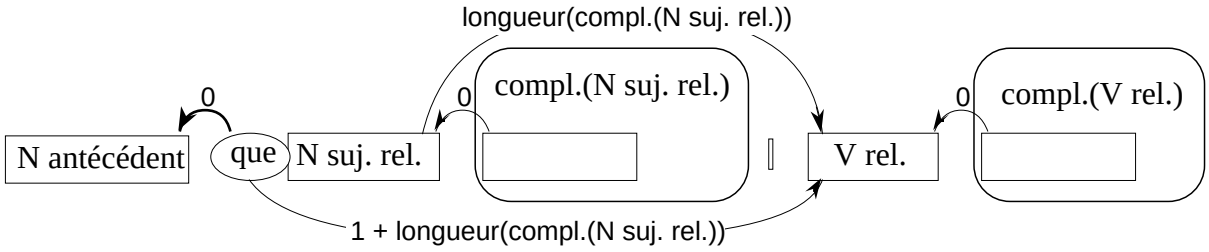
Dans la figure 18, seuls les actants du verbe sont présents, car ses circonstants peuvent être linéarisés après les compléments du sujet quand il est linéarisé après le verbe, comme dans l'exemple de la figure 17, et la linéarisation ne dépend pas de la présence éventuelle de ces circonstants.

Figure 18 :

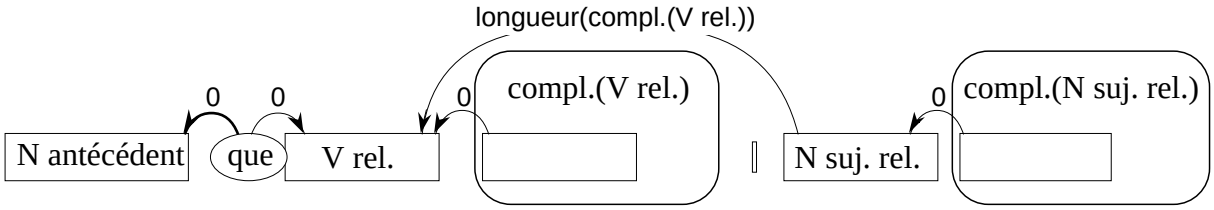
a) arbre de dépendance à transmettre :



b) linéarisation sujet puis verbe (cas de la figure 16)



c) linéarisation verbe puis sujet (cas de la figure 17)



Remarquons que la linéarisation verbe puis sujet place le verbe contigu à son objet *que*, le pronom relatif, et que la linéarisation sujet puis verbe éloigne le verbe de son objet *que*, d'autant plus que le sujet a beaucoup de compléments.

Pour chacune des deux linéarisations, calculons la somme (Σ) des longueurs des dépendances :

- linéarisation sujet puis verbe :

Σ longueurs des dépendances = 1 + 2 * longueur (compléments (N sujet))

- linéarisation verbe puis sujet :

Σ longueurs des dépendances = longueur (compléments (V))

La droite d'équilibre théorique entre les deux linéarisations possibles est la situation d'égalité de ces deux valeurs (voir ci-dessous figure 19) :

1 + 2 ∗ longueur (compléments (N sujet)) = longueur (compléments (V))

Pour que la linéarisation verbe puis sujet soit optimisée, il faut que la somme des longueurs des dépendances soit minimale :

longueur (compléments (V)) ≤ 1 + 2 ∗ longueur (compléments (N sujet))

Les deux linéarisations placent en premier celui du sujet ou du verbe qui a peu ou aucun complément, et ensuite celui qui a beaucoup de compléments.

Les exemples des figures 16 et 17 corroborent cette propriété :

- figure 16 (ordre linéaire sujet puis verbe) : 1 < 8
- figure 17 (ordre linéaire verbe puis sujet) : 0 < 3

Nous présentons maintenant une validation sur corpus de ses propriétés.

Le corpus est composé d'articles du journal Le Monde qui ont servi à l'évaluation des tests du projet GRACE [6], action d'évaluation comparée des étiqueteurs du français. Ce corpus fait 46410 mots et 1886 phrases (fin de phrase par . ? ! ; :), et il a été étiqueté par Josette Lecomte (INALF), linguiste du comité de coordination du projet GRACE, ce qui nous a permis d'extraire les *que* ou *qu'* pronoms relatifs. On compte alors :

65 relatives avec un pronom atone sujet (plus de la moitié)

61 relatives avec un syntagme sujet, parmi lesquelles :

9 relatives dont le pronom relatif sature la valence objet d'un infinitif

2 relatives - citations coupées, d'où une mesure impossible des longueurs

50 relatives restantes, parmi lesquelles :

19 linéarisations sujet puis verbe (38% des 50 relatives restantes).

31 linéarisations verbe puis sujet (62% des 50 relatives restantes)

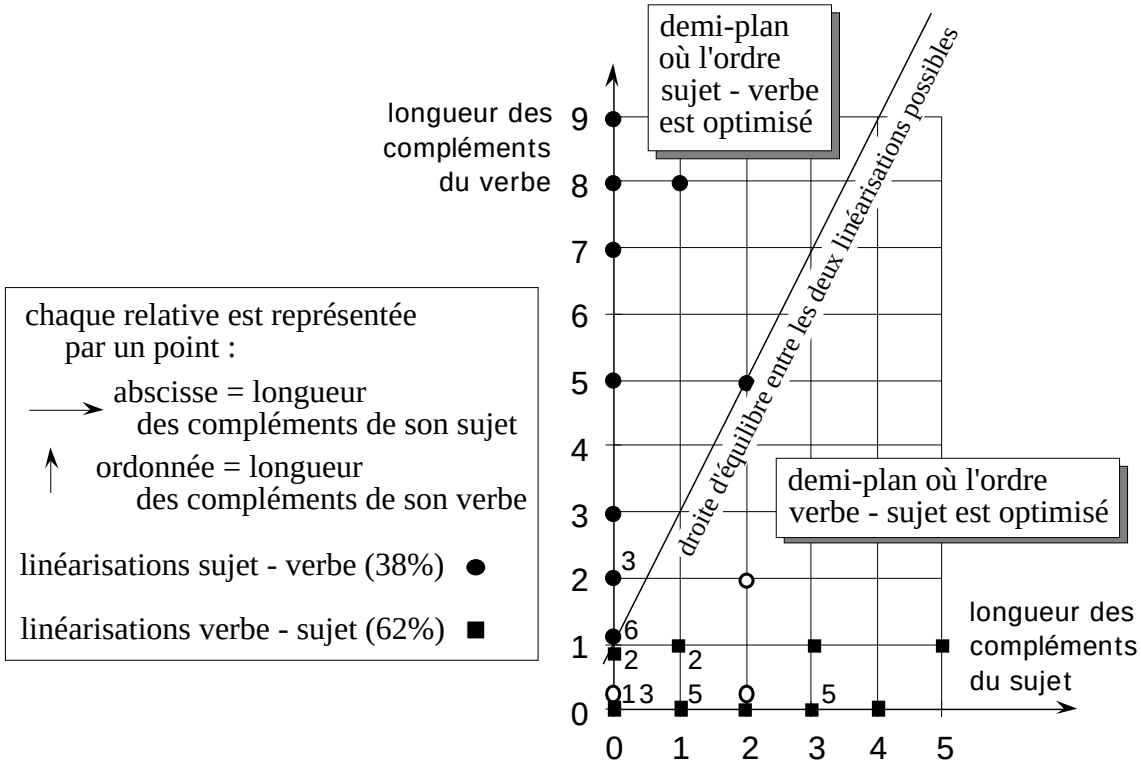
On peut s'étonner de la rareté du phénomène étudié : 50 relatives concernées sur 1886 phrases d'un corpus de 46410 mots. Remarquons aussi que la linéarisation verbe puis sujet est la plus fréquente parmi les relatives où elle est possible (sujet syntagme).

---

[6] Voir la présentation de ce projet (auquel nous avons participé) dans (Paroubek 98) et sur le site : http://limsi.fr/TLP/grace/ (mars 99).

Pour chacune de ces 50 relatives, dans la figure 19, nous avons placé un point qui la représente dans un repère cartésien orthonormé avec la longueur des compléments de son sujet en abscisse et la longueur des compléments de son verbe en ordonnée. Les points sont accompagnés de l'effectif (en italique) si celui-ci est supérieur à 1. De nombreux exemples d'arbre linéarisés de ces phrases sont donnés en annexe A.3 de notre article des Cahiers de Grammaire.

Figure 19 : Repère cartésien longueur (compl. (V)) - longueur (compl. (N sujet))



Le tableau suivant fait le bilan statistique :

type de linéarisation	effectif	dans le demi-plan attendu	sur la droite d'équilibre	autres
sujet - verbe	19	9 (●)	7 (●)	3 (○)
verbe - sujet	31	29 (■)	2 (■)	0
totaux	50	38	9	3

On observe que l'étude théorique est validée sur ce corpus pour 47 relatives sur 50. Trois relatives ne sont pas optimisées, ce sont des linéarisations sujet - verbe, marquées par un petit cercle blanc (○) sur la figure 19. Voici les trois phrases en question, suivies des longueurs des compléments du sujet et du verbe (voir leur arbre linéarisé en annexe A.3.5 de notre article des Cahiers de Grammaire) :

L'OTAN sera à l'Europe ce que l'Organisation des Etats américains (OEA) fut à l'Amérique latine dans les années 60 : un outil de coopération régionale certes, mais fonctionnant de manière inéquitable. (sujet : 2 SNR, verbe : 2 SNR)

De fait, le secteur financier français est l'un de ceux que le commissaire européen à la concurrence, Karel van Miert, connaît le mieux. (sujet : 2 SNR, verbe : 0 SNR)

Une gageure, lorsqu'on pense à la difficulté que la France a eu de faire adopter par ses quatorze partenaires de l'UE au sommet d'Amsterdam, le 18 juin, puis par les chefs des pays les

plus industrialisés, à Denver le 22 juin, de simples déclarations relatives au Proche-Orient.

(sujet : 0 SNR, verbe : 0 SNR)

Dans la première, une contrainte supplémentaire s'est exercée : la structure *X sera à Y ce que A fut à B* impose la linéarisation sujet - verbe. Les deux autres ne sont pas optimisées, probablement à cause une contrainte agissant sur un segment de taille supérieure à la phrase (voir les travaux de Nadine Lucas).

Sur le sujet de "La place du sujet nominal dans les relatives", on pourra consulter utilement (Fuchs 97), dont l'étude sur un numéro entier du journal Le Monde (157 relatives de tout type avec sujet postposable) fait de nombreuses hypothèses (thème-rhème, sémantique, longueur des groupes, …). Nous citons ses observations sur la "longueur du constituant sujet", et sur la "longueur du constituant verbal" (mesurées en syllabes et/ou en mots), page 153 :

> *Si l'on considère la longueur **relative** du groupe sujet par rapport à celle du groupe verbal, on retrouve également la tendance énoncée plus haut : le groupe sujet a tendance à être postposé s'il est plus long que le groupe verbal, et a être antéposé s'il est plus court : c'est donc le plus long des deux groupes qui tend à être postposé.*

Ces observations corroborent globalement les nôtres, et nous pensons leur apporter le fondement théorique de la linéarisation optimisée sous la contrainte du moindre effort de mémoire.

2.3.2 Le processus de réception : phrase linéaire → arbre = mise en relation par l'intermédiaire de mémoires

Nous allons maintenant étudier comment, à la réception, l'arbre de dépendance est reconstruit à partir de l'ordre linéaire, en partant d'une définition dynamique de la mise en relation comme processus fondé sur l'utilisation de la mémoire : souvenir, puis oubli.

2.3.2.1 Définition de la "connexion" chez Tesnière

Souvenons-nous de la définition de la connexion chez Lucien Tesnière dans (Tesnière 59), page 11, § 3 (en fait la première page de Tesnière) :

> *Entre un mot et ses voisins, l'esprit aperçoit des **connexions**, dont l'ensemble forme la charpente de la phrase.*
>
> *Ces connexions ne sont indiquées par rien.*

Dans cette définition, Tesnière présente la connexion comme :

(1) un **processus**	"*aperçoit*"
(2) un **processus** mental	"*l'esprit*"
(3) un **processus** lié à la perception	"*aperçoit*"
(4) un **processus** de calcul	"*indiquées par rien*"

Notons que Tesnière définit sa connexion comme un processus agi par le lecteur - auditeur en situation de **réception**; ceci pose question sur l'ordre structural, qui peut être transformé en ordre linéaire en situation de **production**.

2.3.2.2 Hypothèse sur le processus de mise en relation

Partons de la connexion de Tesnière, définie comme processus de mise en relation en réception, et précisons-la comme un travail du récepteur (lecteur - auditeur) sur sa mémoire, un calcul présent sur la représentation présente d'événements passés (un état présent de la mémoire) :

en lisant - entendant un verbe :

a) le récepteur se souvient du sujet,

b) il relie le verbe au sujet,

c) puis il oublie ce sujet qui a trouvé son verbe, et n'en attend plus.

Le type de cette relation agie est : le récepteur se souvient du sujet. Elle est orientée du verbe vers le sujet, du segment en cours de réception vers le segment dont il se souvient :

sujet <——le récepteur se souvient —— verbe

Nous appelons cette relation intermédiaire : la "**souvenance**", pour la différencier de la dépendance. La "souvenance" est ensuite transformée en dépendance :

sujet ——dépend de——> verbe

Pour se souvenir du sujet, le récepteur l'a auparavant mémorisé:

en lisant - entendant un syntagme nominal :

le récepteur le mémorise comme sujet potentiel

Nous proposons alors la définition suivante :

la souvenance est un processus et le résultat de ce processus, **processus** de mise en relation de 2 SNR au cours de la réception par l'auditeur - lecteur, **processus** de calcul fondé sur sa mémoire, en deux temps distincts : temps de la réception du sujet potentiel, temps de la réception du verbe.

Remarquons que l'on trouve un concept analogue de processus en 2 temps dans (Grunig 93), page 15 :

si un adjectif a *est suspendu depuis le temps* t *en attente d'un* n *et qu'il en rencontre un à un temps* t', *l'établissement en ce temps* t' *de la connexion entre* a *et* n *peut constituer le signal mettant un terme à la suspension.*

2.3.2.3 Le processus de mise en relation en deux temps modélisé sur ordinateur

Nous allons maintenant décrire la modélisation sur ordinateur de ce processus de mise en relation, puis sa généralisation à tous types de relation, et enfin sa validation par son intégration dans un analyseur automatique et par l'analyse d'un corpus de taille importante.

Présentation du modèle : cas de la mise en relation d'un verbe avec son sujet

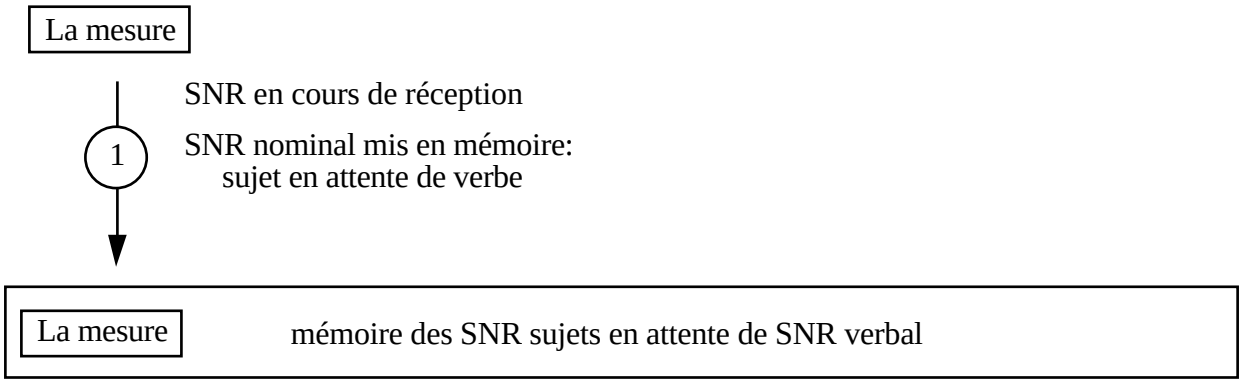
Nous partons de l'hypothèse que cette relation de souvenance est établie par un processus en deux temps, par l'intermédiaire de mémoires spécialisées par type de relation.

Nous allons reprendre l'exemple de la relation entre un SNR nominal sujet et un SNR verbal, relation qui, au cours de la réception, sera établie en deux temps distincts et successifs :

- **temps 1** : un SNR nominal est reçu, puis mémorisé comme sujet potentiel, en attente d'un SNR verbal éventuel (voir figure 20, ci-dessous),

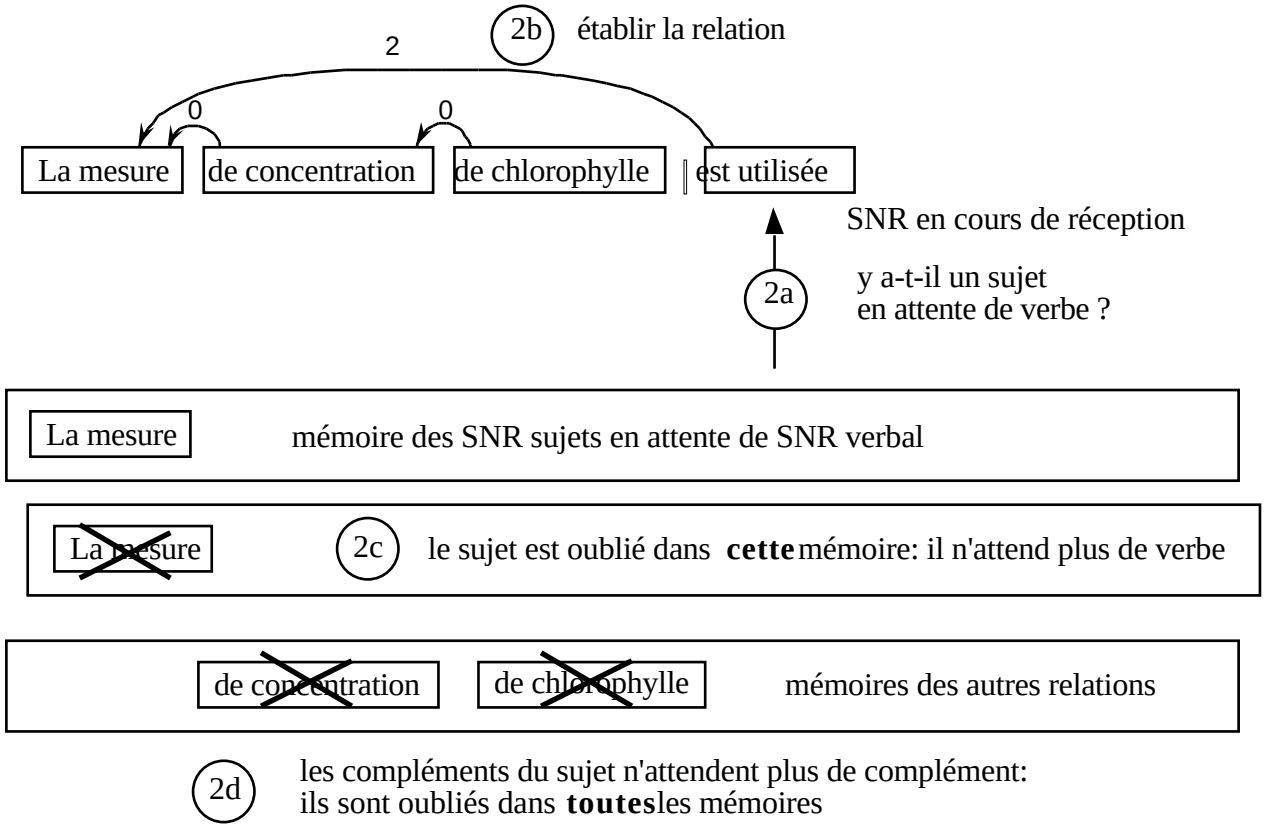
- **temps 2** : puis, si un SNR verbal arrive (est reçu), il est relié au SNR nominal en attente, qui alors n'attend plus de SNR verbal, et est donc "oublié" dans la mémoire des sujets en attente de verbe (voir figure 21, ci-dessous).

Figure 20 : Temps **1** du processus de mise en relation entre deux SNR



Entre le temps 1 et le temps 2 de cette mise en relation sujet - verbe, les deux SNR nominaux dépendant du sujet lui sont reliés par des processus identiques, mais par l'intermédiaire de la mémoire des régissants nominaux en attente de dépendants nominaux.

Figure 21 : Temps **2** du processus de mise en relation entre deux SNR



Le temps 2 comporte quatre opérations :

- 2a : la mémoire des sujets en attente de verbe est consultée,
- 2b : la relation est établie entre le verbe et le sujet en attente ,
- 2c : le sujet n'attend plus de verbe : il est oublié de cette mémoire,
- 2d : tous les compléments du sujet sont oubliés de toutes les mémoires, car ils n'attendent plus de compléments : l'arrivée du verbe clôt définitivement la chaîne des compléments du sujet.

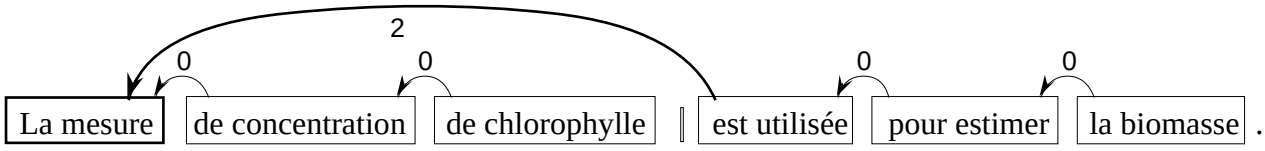
C'est la mise en relation en deux temps distincts, et plus particulièrement cette opération 2d, qui permet et concrétise l'interdépendance des différents processus de mise en relation en cours simultanément durant la réception. L'ensemble de ces processus interdépendants de mise en relation aboutit à l'arbre linéarisé des relations de souvenance de la figure 22 a).

Arbre de dépendance reconstruit par les processus de mise en relation

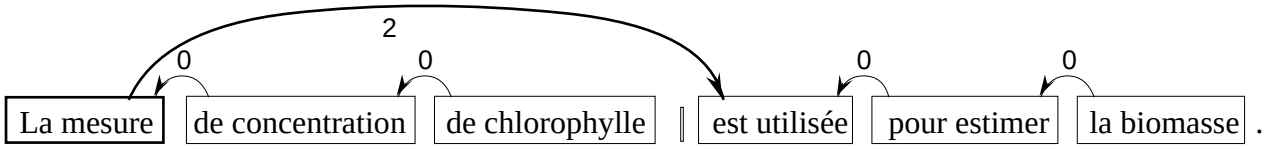
La figure 22 montre les trois étapes finales de la modélisation de la reconstruction de l'arbre de dépendance reçu par l'auditeur - lecteur. En particulier, les relations de dépendance sont restituées à partir des relations de souvenance, ce qui consiste à inverser les relations sujet - verbe. Enfin l'arbre est extrait de la linéarité, d'où la disparition de la métrique, et de toute trace écrite de la prosodie (ponctuation, groupes prosodiques, coupures prosodiques) .

Figure 22 : Les trois étapes finales de la modélisation de la reconstruction de l'arbre de dépendance reçu par l'auditeur - lecteur

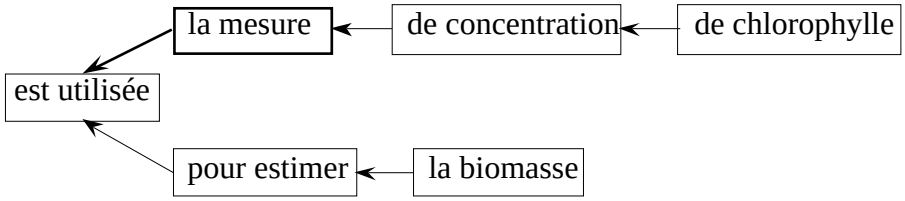
a) arbre linéarisé des relations de souvenance :



b) arbre de dépendance linéarisé, calculé à partir de l'arbre des relations de souvenance



c) arbre de dépendance reconstruit et reçu par l'auditeur - lecteur :



Généralisation du processus de mise en relation par l'intermédiaire de mémoires spécialisées pour chaque type de relation

Ce processus en deux temps est généralisé à l'aide de mémoires spécialisées, une pour chaque type de relation, ce qui organise et cloisonne les différents types d'attentes :

- mémoire des SNR sujets	en attente	d'un SNR verbe
- mémoire des SNR verbes transitifs	en attente	d'un SNR objet
- mémoire des SNR	en attente	d'un SNR subordonné
- mémoire des SNR	en attente	d'un SNR coordonné
- mémoire des SNR antéposés	en attente	d'un SNR régissant
- mémoire des "*que*" pronom relatif	en attente	d'un SNR verbal transitif
- mémoire des "*ne*"	en attente	de "*que*", "*ni*"
- …	(les attendus arrivent ou n'arrivent pas)	

Chaque mise en relation **interagit** avec les autres mises en relation en cours en provoquant l'**oubli** des SNR en attente entre les 2 SNR reliés. C'est la seule interaction entre les différentes mémoires. Dans son état actuel, le modèle efface toutes les mémoires en fin de phrase, ce qui entraîne qu'aucune relation n'est établie entre deux phrases.

Notons qu'aucune hypothèse n'est faite explicitement sur les structures syntaxiques situées entre les deux segments reliés, ni sur la distance qui sépare ces deux segments.

L'analyseur syntaxique automatique, cadre de la modélisation du processus de mise en relation

Ce processus est actuellement modélisé dans le cadre d'un analyseur syntaxique automatique qui produit l'arbre de dépendance sous la forme de la figure 22, avec les relations de coordination et d'antécédence. Cet analyseur est décrit dans (Giguet-Vergne 97), (Vergne-Giguet 98) et (Giguet 98), et globalement en section 3.3. du présent mémoire.

Pour les valider, ces concepts ont été confrontés avec des corpus variés en français (journal Le Monde, littérature, textes scientifiques) et de taille importante (corpus GRACE = 650 000 mots), au moyen de l'analyseur syntaxique. L'efficacité opératoire est déjà importante [7] et valide le principe de la mise en relation en deux temps par l'intermédiaire de mémoires spécialisées par type de relation [8]. L'ordinateur étant une machine à exécuter chronologiquement des actions, son utilisation comme outil de modélisation nous focalise en priorité sur l'explicitation des processus plutôt que sur l'explicitation des structures, et c'est ce qui nous a amené à concevoir ce processus de mise en relation, et à l'étendre ensuite à la définition de la souvenance.

2.4. Conclusion

Dans cette partie, nous avons tenté de présenter en un ensemble cohérent les deux processus de production et de réception qui permettent à un humain de transmettre un arbre de dépendance à un autre humain, associés aux deux formes structurelles que peut prendre une phrase : son ordre linéaire et son arbre de dépendance, arbre codé, compressé temporairement dans l'ordre linéaire; et nous avons mis en évidence les contraintes géométriques, informationnelles, chronologiques et mémorielles qui façonnent ces deux processus et ces deux structures.

[7] Voir les corpus analysés sur internet à l'adresse : http://www.info.unicaen.fr/~giguet

[8] Que nous apprend cette validation par analyse syntaxique automatique d'un corpus ? On constate la bonne capacité opératoire de l'analyseur, on constate la faisabilité de la généralisation des processus. On peut ensuite s'interroger sur la validité psycholinguistique de ces processus. Ceci reste une conjecture ouverte : seules des expériences réalisées par des psycholinguistes pourront apporter un début de réponse.

3. Informatique linguistique :

Analyse syntaxique automatique non combinatoire et de complexité linéaire sur corpus

Dans cette partie, je présente une synthèse de mes travaux en analyse syntaxique automatique, que l'on pourrait résumer comme une progression des processus combinatoires vers les processus non combinatoires. Après une présentation de cette progression (en 3.0.), je caractérise les bases linguistiques et algorithmiques des analyseurs classiques pour tenter de comprendre leur aspect combinatoire (en 3.1.), puis (en 3.2.) le "tagging" est présenté comme premier processus d'analyse syntaxique non combinatoire, grâce à l'exploitation du contexte; enfin (en 3.3. et 3.4.) je présente deux réalisations originales : un analyseur non combinatoire de phrase (l'analyseur 98, non combinatoire depuis 93), et le passage de l'analyse de phrase à l'analyse de flux textuel (l'analyseur 99).

3.0. Analyse syntaxique automatique : des processus combinatoires vers des processus non combinatoires, ou du choix vers le calcul

Donnons tout d'abord notre définition d'un processus combinatoire : un processus combinatoire est un processus qui **choisit** des valeurs d'attributs d'objets parmi des valeurs possibles explicitées a priori : dans l'analyse syntaxique, il doit choisir les catégories des mots; mais le processus n'a aucun critère pour faire directement un *choix local* : ce mot a cette catégorie; il a seulement des critères pour faire un *diagnostic global* sur la solution complète : cette suite de mots est ou n'est pas une phrase selon un ensemble d'étiquettes de mots et une grammaire formelle. Il faut donc énumérer toute la combinatoire des valeurs des attributs des objets : les *choix locaux* sont essayés systématiquement dans un ordre quelconque ou heuristique, et si une suite de choix aboutit à un échec (diagnostic global d'échec), une autre suite de choix est essayée (ceci est habituellement appelé le retour en arrière), mais on ne sait pas quel choix est erroné; ceci implique un processus arborescent au cours duquel aucune ou plusieurs solutions sont trouvées, processus de complexité théoriquement exponentielle. Notons que les processus combinatoires se situent dans le paradigme de l'ordinateur comme machine à choisir (et non comme machine à calculer - cf. ci-dessous en 3.2.2.3), avec l'hypothèse que les valeurs possibles des attributs sont explicitables, et donc qu'une langue est fermée.

Les années 57 à 97 constituent 40 ans de suprématie du paradigme de la compilation en TAL syntaxique, paradigme fondé sur l'hypothèse qu'une langue peut être traitée et modélisée de la même manière qu'un langage de programmation. Cette hypothèse, d'abord implicite, puis explicitée par Bernard Vauquois en 1960 après sa participation au groupe ALGOL, est manifestement erronée, car un langage de programmation est exhaustivement caractérisé, et une langue est ouverte et très partiellement

caractérisée, mais cette hypothèse n'est toujours pas explicitement pointée dans les publications du domaine comme cause majeure des échecs des TAL syntaxiques. Une explication possible en est l'absence d'un cadre théorique de substitution (cadre que nous pensons apporter), et la difficulté de la communauté à opérer cette mutation fondamentale, qui va entraîner le difficile abandon des grammaires formelles comme outil de modélisation des langues.

Mais le paradigme de la compilation est, de fait, maintenant progressivement abandonné sous la pression du "tagging", qui substitue à l'explicitation des structures l'explicitation des processus de déduction fondés sur le contexte, et qui du point de vue fonctionnel, rend une partie des services attendus de l'analyse syntaxique (l'étiquetage des "mots" sans mise en relation).

Du point de vue algorithmique, observons que la compilation est de complexité linéaire, mais que la compilation transposée en TAL conduit à des algorithmes combinatoires. Pourquoi ? Cette question est rarement posée. La cause vient du fait des différences radicales des deux codes analysés : tout est connu d'un langage de programmation, une langue est partiellement connue; le processus d'analyse est incapable de faire des choix locaux sur des critères locaux, et les algorithmes sont de complexité théorique exponentielle, et de complexité pratique au mieux en $O(n^2)$; alors que le tagging a une complexité linéaire, car des règles de déduction contextuelle en nombre constant sont appliquées à un token et ses voisins.

Sur le plan linguistique, en reprenant les concepts de Ferdinand de Saussure, cette mutation consiste à passer d'un modèle de la "langue" à un modèle de la "parole" : on ne traite plus un idéal de langue, mais des corpus attestés tous venants.

Au plan informatique, l'ordinateur était considéré comme une machine à stocker a priori des résultats attendus (des mots étiquetés et des structures), et à choisir dans le stock, et il est maintenant plus employé pour ce qu'il est : une machine à calculer des résultats à partir de données. De machine à reconnaître, il devient machine à connaître, connaître en tant que calcul.

Sur le plan des industries des langues, la complexité non linéaire des algorithmes limite les applications au traitement des "phrases" très courtes, du type de celles qui sont tapées par un utilisateur dans un système d'interrogation de base de données en texte libre, comme l'application dite des pages jaunes (ERLI et France Télécom). Pour des phrases plus longues, un algorithme combinatoire doit les traiter en différé, avec une durée d'analyse non prévisible, et une absence possible de résultat, mais de nombreuses applications, telle la synthèse vocale, nécessitent un traitement en flux, à débit constant, ce qui rend la complexité linéaire obligatoire. C'est le cas de notre analyseur, déjà intégré dans plusieurs projets industriels (voir ci-dessous en 4.2 et 4.3).

3.1 Les analyseurs combinatoires traditionnels

Cette section est une synthèse partielle de "Les cadres théoriques des TAL syntaxiques : quelle adéquation linguistique et algorithmique? une étude et une alternative", TALN'95, Marseille, juin 1995. Cet article a été écrit en réaction à l'appel à communication pour la table ronde "TALN et Linguistique" : "On dispose ainsi, notamment pour le français, de descriptions effectuées dans des cadres théoriques appropriés et donc utilisables dans les systèmes de TALN."

3.1.1 Bases linguistiques traditionnelles

Cadres théoriques linguistiques actuels des TAL syntaxiques

Ils sont presque exclusivement issus des travaux de Noam Chomsky, importés en TAL de la linguistique théorique. Rappelons les grandes lignes du programme chomskien :

- fonder une théorie de la compétence du locuteur natif, dans le cadre d'une épistémologie popperienne (voir aussi en 5.2);
- les règles syntagmatiques de génération sont à la fois les hypothèses et l'outil de déduction dans la démarche hypothético-déductive : la génération par réécriture constitue le processus de déduction;
- les règles syntagmatiques modélisent des structures profondes et non des structures de surface;
- on reste dans le cadre de la phrase, le plus souvent simple (proposition principale, éventuellement une subordonnée), sans grand rapport (en complexité et longueur) avec les phrases réelles des textes : en fait des propositions artificielles, abstraites, expérimentales, produites par la génération;
- étant donné une grammaire générative, une phrase qui peut ou ne peut pas être générée, est dite grammaticale ou agrammaticale ;
- recherche fondamentale pure sans préoccupation de TAL, ni TAL à but opératoire, ni TAL à but de confrontation entre concepts et objet réel.

Ces cadres théoriques se sont concrétisés dans divers formalismes, tels que GPSG, HPSG, LFG, TAG, tous issus du même moule théorique.

D'autre part, on note la légère influence de Lucien Tesnière : les grammaires et les arbres de dépendance ont été importés dans les TAL aux débuts de la traduction automatique (cf. [Tesnière 59]), car la langue source était le russe, et Lucien Tesnière avait écrit des grammaires du russe qui furent utilisées au CETA (puis GETA) à Grenoble.

La nécessité de théoriser explicitement les relations de dépendances est apparue à cause des lacunes des grammaires syntagmatiques, centrées sur la description explicite des constituants et qui laissent dans l'implicite la description des relations.

Ces deux cadres théoriques ont été traditionnellement opposés, alors qu'il est nécessaire de concevoir un cadre théorique où segmentation et relation sont réunies.

Inadéquation de ces cadres théoriques linguistiques aux TAL syntaxiques

Nous allons tenter de montrer leur inadéquation aux TAL syntaxiques :

	cadres linguistiques	TAL syntaxiques
différence des objets :	compétence	langue concrète (≠ performance)
=> travail sur corpus :	aucun	indispensable
différence des objectifs :	modélisation théorique d'un objet statique	objectif opératoire = processus (dynamique)
opposition des démarches :	génération : phrase sortie	analyse : phrase en entrée

L'objet traité en TAL n'est pas la compétence ("la **connaissance** que le locuteur-auditeur a de sa langue", cf. [Chomsky 71], page 13), ni la performance ("l'**emploi** effectif d'une langue dans des situations concrètes", cf. [Chomsky 71], page 13), mais la part concrète, matérielle d'une langue : texte ou signal de parole en tant qu'**objet concret, matériel**, extérieur à l'humain.

Les grammaires formelles constituent un outil adapté à la modélisation de la syntaxe d'un langage de programmation, à la conception de compilateur, à la description d'un langage formel. Mais c'est un outil inadapté à la description d'une langue en tant qu'objet concret, extérieur à l'humain, car une langue concrète a très peu de caractéristiques communes avec un langage formel (comme un langage de programmation). Ce point central est développé en 5.1.4.

Développons les aspects liés à la syntaxe :

La redondance des formes est une caractéristique des langues, comme de tout code utilisé par des êtres vivants (code génétique par exemple); elle permet une robustesse de la transmission et de la mémorisation des informations; comme un langage formel n'est pas redondant, une grammaire formelle n'est pas appropriée à tirer parti de cette redondance, qui constitue pourtant un des fondements des TAL.

La récursivité des segments (et donc des règles) est une hypothèse sur les structures profondes de la compétence du locuteur natif, mais elle n'est pas indispensable pour modéliser la syntaxe des langues concrètes, car il n'y a jamais une infinité de compléments, ni des insertions multiples illimitées, alors qu'elle est indispensable pour modéliser la syntaxe d'un langage de programmation, car il n'y a pas de limite a priori à l'enchâssement des instructions.

La polycatégorie (inexistante dans les langages formels), ou le fait qu'une même graphie recouvre plusieurs rôles syntaxiques et plusieurs sens est une conséquence de la correspondance forme-sens non biunivoque dans les langues; c'est un phénomène lexical hors-contexte, du point de vue du dictionnaire, mais il n'a plus d'existence en contexte.

Ce qu'on appelle d'habitude "ambiguïté" est en fait l'incapacité de choisir pour la machine qui analyse (en accord avec [Rady 83]), à cause du manque d'informations sur l'objet traité, du manque de contexte, mais *ce n'est pas une propriété intrinsèque de l'objet traité* (contrairement à [Rady 83]) : cette "ambiguïté" est un artefact dû à l'absence de contexte, et cette incapacité de choisir entraîne une combinatoire artificielle; réservons plutôt à "ambiguïté" le sens d'une hésitation entre plusieurs interprétations pour l'humain qui écoute ou lit.

Quand on assigne à un analyseur l'objectif d'accepter ou de refuser une phrase pour diagnostiquer sa grammaticalité selon une grammaire formelle, on le place dans la position du "locuteur natif" qui juge de l'acceptabilité d'une phrase générée : ce n'est pas le rôle qu'on attend d'un analyseur.

Assignons plutôt à un analyseur l'objectif de produire une analyse pour toute phrase entrée.

On constate enfin l'absence de cadre théorique pour décrire et modéliser (et traiter) les formes des phrases de complexité syntaxique et de longueur telles qu'on les observe dans des textes attestés (avec des propositions subordonnées, antéposées ou incises, des verbes antéposés, des coordinations, des subordinations nombreuses et variées).

3.1.2 Bases algorithmiques traditionnelles

Inadéquation algorithmique des cadres théoriques actuels des TAL

On peut observer deux types de causes d'inadéquation algorithmique :

- les causes externes : l'inadéquation linguistique implique une inadéquation algorithmique : les algorithmes utilisent peu et mal les propriétés de l'objet réellement traité (propriétés des langues concrètes)
- les causes internes : les algorithmes fonctionnent par analogie avec la compilation, et sont fondés sur les propriétés (connues exhaustivement) des langages formels, alors qu'on traite des langues concrètes (propriétés connues partiellement)

Ceci entraîne que les algorithmes sont combinatoires : des *choix locaux* sont faits (presque) à l'aveuglette, car avec peu de critères de choix (ces critères ne peuvent être que d'origine linguistique); on a uniquement un critère de *diagnostic global* en fin d'analyse : cette chaîne est ou n'est pas une phrase, selon la grammaire formelle.

Voici les causes de la combinatoire des algorithmes, c'est-à-dire les caractéristiques des choix locaux:

- polycatégorie (inexistante dans les langages formels) => algorithme combinatoire, car on n'a pas de critère pour choisir la catégorie (propriété de langue inexistante dans un langage formel);
- syntagme récursif (non motivé dans les langues concrètes) => algorithme combinatoire, car, le syntagme incluant ses compléments, on segmente et on relie simultanément, sans critère pour borner le syntagme ni pour choisir les rattachements (relation = propriété de langue inexistante dans un langage formel)
- grammaire formelle => algorithme combinatoire, car plusieurs règles sont applicables simultanément, sans critère pour choisir quelle règle
- syntagme récursif dans une grammaire formelle => algorithme combinatoire, car on n'a pas de critère pour choisir combien de fois appliquer une règle récursive
- pas (ou peu) d'exploitation du contexte (règles hors contexte), ni de la redondance des formes.

Un algorithme combinatoire implique un processus arborescent : à chaque nœud, essai d'un choix-branche, échec éventuel, diagnostiqué à une feuille, => retour en arrière, pour essayer les autres branches de chaque nœud; l'algorithme est de complexité théorique exponentielle en temps, et de complexité pratique au mieux quadratique.

Un algorithme combinatoire implique la sortie de zéro ou plusieurs analyses sans critère pour choisir.

Notons qu'une grammaire formelle utilisée en analyse syntaxique automatique remplit simultanément deux fonctions : elle guide le processus d'analyse par l'application des règles (par réécriture en sens inverse de la génération), et elle code les structures des syntagmes (dans le membre droit des règles).

Enfin, on fait l'hypothèse implicite mais erronée que tout l'objet analysé est connu (comme pour un langage formel) : tous les mots, toutes leurs catégories possibles, toutes les structures : ces attendus sont irréalistes et imposent des rattrapages par des procédures ad hoc (qui exploitent enfin contexte et redondance).

Point de vue historique : la fin des grammaires formelles comme outil de modélisation des langues ?

Pourquoi les grammaires formelles ont-elles pris une telle importance en TAL?

Chomsky a eu une formation initiale en mathématique, puis il a importé en linguistique son bagage mathématique : les grammaires formelles sont nées en linguistique, sur un substrat mathématique.

ALGOL 60 est le premier langage de programmation décrit par une grammaire formelle.

Dans les années 60, Bernard Vauquois (qui faisait partie des créateurs d'ALGOL) fonde la traduction automatique *de langues* de seconde génération sur la compilation, traduction automatique *de langages formels* (voir [Boitet 92] page 50).

Du point de vue historique, le TAL reçoit donc les grammaires formelles de 3 origines différentes : la syntaxe chomskienne (linguistique), la compilation (informatique), et la traduction automatique (pratique fondatrice du TAL).

Ceci permet de comprendre pourquoi les grammaires formelles ont un rôle central en TAL depuis 40 ans, et pourquoi la communauté du TAL a tant de difficultés à les abandonner en tant qu'outil de modélisation des langues.

3.2. Le "tagging" : processus non combinatoire par exploitation du contexte

Cette section est une synthèse de notre article "Regards Théoriques sur le «Tagging»", TALN'98, Paris, juin 1998, écrit pour prendre du recul par rapport au "tagging" comme pratique et technique destinées à pallier les limites et les blocages de l'analyse syntaxique combinatoire.

Le "tagging", ou "étiquetage", ou "marquage", consiste à affecter une "étiquette" ("tag", ou catégorie) à chaque "mot" d'un texte.

Le tagging constitue la passerelle entre l'analyse syntaxique combinatoire et l'analyse syntaxique non combinatoire, et prépare dans les faits l'abandon des bases de l'analyse syntaxique combinatoire : les grammaires formelles et le syntagme récursif.

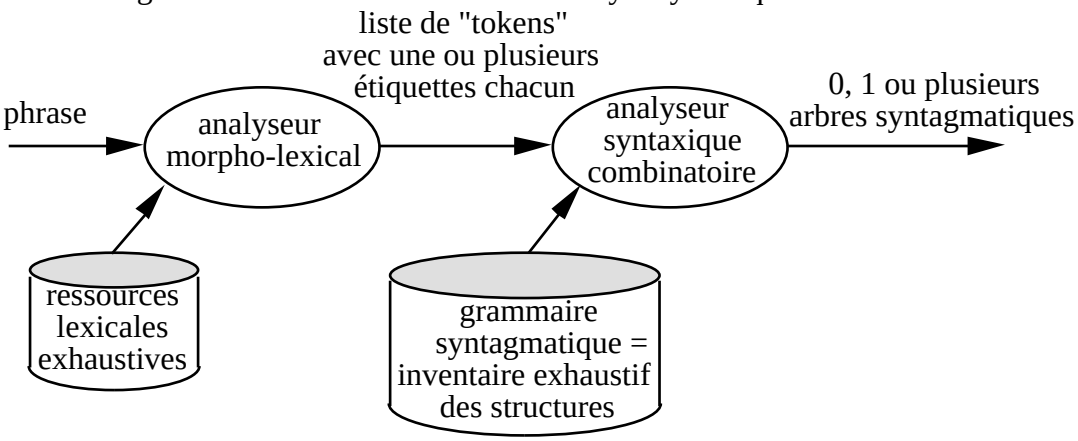
De plus, dans le tagging, le rôle de l'ordinateur évolue : de la "désambiguïsation" comme choix parmi des valeurs explicites a priori, au calcul de valeurs résultats à partir de valeurs de départ.

3.2.1. Tagging et analyse syntaxique traditionnelle

3.2.1.1. Deux opérations complémentaires dans une chaîne de traitement

Par "analyse syntaxique traditionnelle", nous entendons l'analyse syntaxique dans son état canonique telle que la décrit Gérard Sabah dans le chapitre 2 de (Sabah 1989), pages 37 à 71 : "Les principes de l'analyse syntaxique des phrases".

Figure 1 : Chaîne de traitement de l'analyse syntaxique traditionnelle

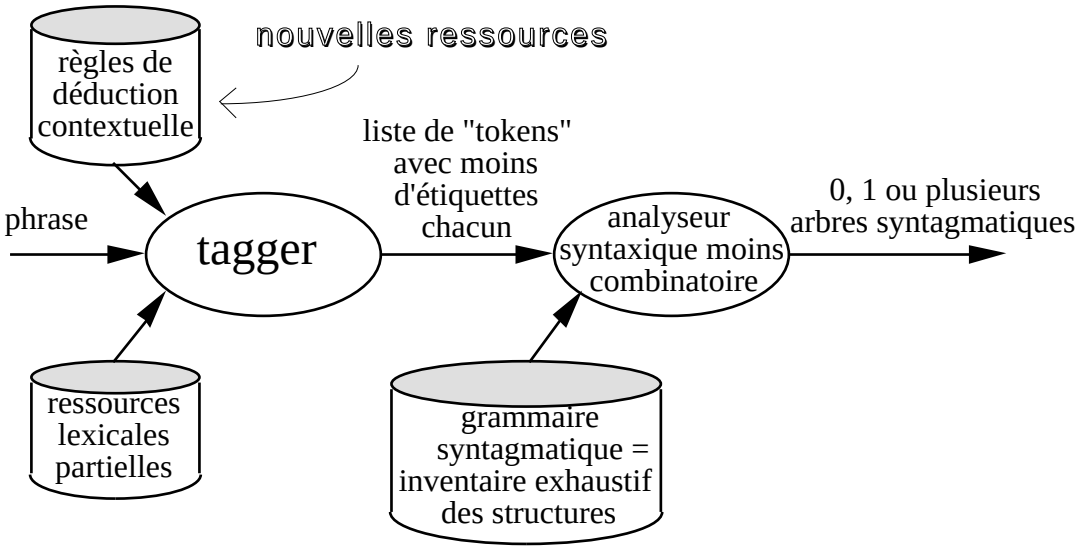


La chaîne de traitement de l'analyse syntaxique traditionnelle (voir figure 1 ci-dessus) comprend au minimum deux modules : l'analyseur morpho-lexical, qui, à partir de ressources lexicales considérées comme exhaustives, produit une liste de "tokens" ("mots" arbitraires) munis chacun d'une ou plusieurs étiquettes, suivi de l'analyseur syntaxique proprement dit, qui, pour chaque phrase, produit zéro, un ou plusieurs arbres syntagmatiques.

Le problème central de l'analyse syntaxique traditionnelle est son aspect combinatoire : l'analyseur doit choisir une étiquette pour chaque "token", et l'ensemble des étiquettes choisies (par les substitutions lexicales) doit permettre d'attribuer à la phrase une structure syntaxique attendue, reconnue dans l'inventaire exhaustif des structures attendues, codé sous la forme d'une grammaire syntagmatique. La cause profonde de cet aspect combinatoire est que l'analyse syntaxique traditionnelle est fondée sur les principes de la compilation, analyse d'un langage formel dont le lexique est clos (les mots ayant une seule catégorie) et dont la syntaxe est exhaustivement définie par une grammaire formelle, alors qu'une langue a un lexique ouvert (les mots ayant plusieurs catégories) et une syntaxe partiellement définie.

Dans la nouvelle chaîne de traitement, le tagger vient se substituer à l'analyseur morpho-lexical, pour fournir à l'analyseur syntaxique une liste de "tokens" munis chacun d'une seule étiquette (ou le moins possible), et ainsi annuler, ou réduire le plus possible, la combinatoire des catégories possibles, par l'apport de nouvelles ressources : les déductions contextuelles (voir figure 2 ci-dessous); mais l'analyseur syntaxique est inchangé.

Figure 2 : Nouvelle chaîne de traitement de l'analyse syntaxique traditionnelle



3.2.1.2. Deux démarches opposées vis-à-vis des structures et des processus

Résumons ces oppositions dans le tableau suivant :

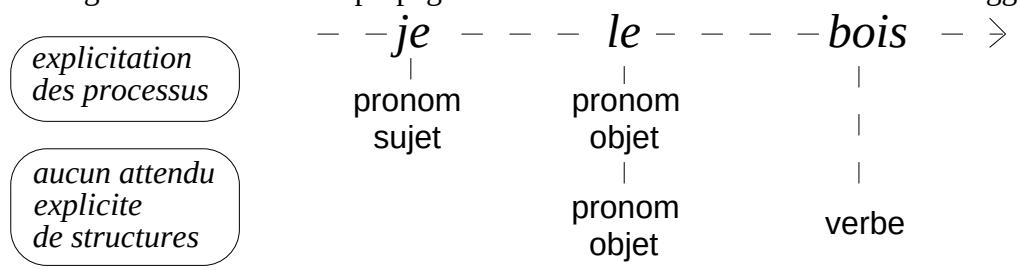
	analyse syntaxique traditionnelle	*tagging*
structures	explicitées exhaustivement	non explicitées
processus	analogue à la compilation, mais combinatoire	explicité par les déductions contextuelles

Dans l'analyse syntaxique traditionnelle, l'ensemble des structures syntaxiques attendues est exhaustivement explicité sous la forme d'une grammaire syntagmatique.

À l'opposé, le tagging privilégie le processus par rapport aux structures. Le processus du tagging est une propagation de déductions contextuelles sur les tokens (voir figure 3 ci-dessous). Le processus est conduit par les fréquences des contiguïtés des catégories dans un tagger statistique, ou explicité par des règles contextuelles symboliques dans un tagger symbolique, mais aucune structure n'est attendue explicitement. L'algorithme d'un tagger consiste à tester sur chaque token l'applicabilité d'un nombre constant de règles, et ceci lui confère une complexité linéaire en temps par rapport au nombre de tokens de la phrase.

Dans un renversement complet par rapport à l'analyse syntaxique traditionnelle, le tagging explicite le processus mais n'explicite pas les structures.

Figure 3 : Processus de propagation de déductions contextuelles dans le tagging



Du point de vue de son processus, un analyseur traditionnel intègre toutes les caractéristiques du modèle de la compilation, à la différence non négligeable qu'un compilateur analyse un langage de programmation, exhaustivement connu et modélisé, et qu'un analyseur syntaxique de langue analyse une langue, connue partiellement, et donc partiellement modélisée. En conséquence, alors que le processus d'un compilateur est déterministe, celui d'un analyseur syntaxique traditionnel de langue est non déterministe, car l'absence de critères locaux pour prendre des décisions locales implique des retours en arrière aux tokens précédents (voir les "points d'embarras" de Sabah 1989, page 58). Les algorithmes sont de complexité polynomiale avec des "formalismes simples", ou bien le problème devient NP-complet "pour les formalismes plus évolués, comme les grammaires d'unification" (voir Abeillé et Blache 1997, pages 79 et 80).

3.2.2. Tagger = "désambiguïser" ou calculer ?

3.2.2.1 La "désambiguïsation" classique

Le tagging est conçu classiquement comme une "désambiguïsation" (comme dans l'action GRACE, voir ci-dessous en 4.1), c'est-à-dire comme le choix automatique d'une étiquette parmi toutes les étiquettes possibles explicitées, énumérées, comme une annulation ou une diminution de l'"ambiguïté". La désambiguïsation pose mal le problème du tagging, car tout le possible n'est pas énumérable, n'est pas explicitable a priori hors contexte, car tous les tokens possibles ne sont pas énumérables, et toutes les étiquettes possibles d'un token donné ne sont pas énumérables. Autrement dit, une langue est ouverte, et la "désambiguïsation" suppose sa fermeture (langue fermée comme un langage de programmation).

La "désambiguïsation" reste dans la continuité du paradigme de la compilation, et constitue le type classique de collaboration du lexical et du contexte dans le tagging : tout est explicité dans le lexique, qui constitue la base de départ : le lexique est supposé exhaustif, et est filtré ensuite par le contexte.

De plus, la "désambiguïsation" classique consiste en un choix parmi l'ensemble des étiquettes complètes, indivisibles, avec toute la combinatoire des attributs (311 étiquettes pour GRACE), ce qui interdit des déductions sur les attributs, indépendamment les uns des autres.

3.2.2.2 Critique de la "désambiguïsation"

L'ambiguïté est définie différemment en linguistique et en TAL. L'ambiguïté est ainsi définie en linguistique :

− ambiguïté lexicale : un mot a intrinsèquement plusieurs *sens* possibles,

− ambiguïté phrastique : une phrase a plusieurs *sens* possibles pour l'humain récepteur (cf. Catherine Fuchs 1987 p15).

L'ambiguïté est ainsi définie dans le TAL : un mot a plusieurs *catégories* possibles hors contexte; en contexte, il en prend une; cette polycatégorie n'existe que hors contexte; c'est une apparence lexicale; en contexte, il n'y a plus de polycatégorie; il suffit de sortir de la chronologie d'analyse : lexique d'abord, contexte ensuite. Un concept étant différent de la réalité, on peut l'abandonner; le concept de désambiguïsation est doublement à abandonner : il n'est pas lié à l'ambiguïté linguistique, et la polycatégorie est un artefact dû à l'absence de contexte.

L'idée couramment répandue qu'il faut "laisser de l'ambiguïté" pour la suite du processus vient du paradigme de la "désambiguïsation"; elle est erronée pour les raisons suivantes : il n'est pas possible d'énumérer tout le possible; la suite du processus n'a pas à choisir parmi du possible énuméré, mais peut modifier des valeurs des attributs; si la "bonne étiquette" n'est pas dans la liste, cela n'interdit pas à la suite du processus de la calculer (par le contexte ou la mise en relation); une valeur indéterminée est une valeur, un processus calcule ce qu'il peut calculer, et le processus suivant continuera le calcul et pourra modifier des valeurs des attributs, indépendamment les uns des autres.

Le concept de "désambiguïsation" est donc à abandonner.

3.2.2.3 Après la "désambiguïsation" : le calcul

Prenons la métaphore de la résolution d'équation : c'est un calcul à partir de données, et de contraintes, le résultat ne préexiste pas explicite dans les données. La solution de l'équation n'est pas à choisir dans une ressource statique contenant a priori toutes les solutions, mais à calculer à partir des données.

Dans un traitement de langue, il faut trouver un processus qui prend l'ouverture comme réelle dès la conception du processus, avec un autre type de collaboration du lexical et du contexte : des ressources lexicales partielles constituent une amorce des déductions contextuelles. Le contexte ne filtre pas le lexical supposé exhaustif, mais comble les lacunes du lexical supposé partiel.

Dans le processus de calcul, les attributs des tokens ont des valeurs initiales, qui évoluent au long du calcul, à partir de 2 types de ressources : les ressources lexicales + le contexte. Par exemple, pour "le" : la valeur initiale est déterminant masculin singulier (du lexique des mots grammaticaux); le rôle du contexte est défini par une règle contextuelle : "le" + verbe => "le" pronom atone objet.

Les attributs ont au départ une valeur par défaut : la valeur par défaut est profondément inscrite dans la langue, sans doute une économie, une souplesse, une ouverture cognitive. En abandonnant la "désambiguïsation", on abandonne l'étiquette comme indivisible, une étiquette n'est pas un atome, indivisible, qu'il faudrait choisir comme un tout parmi une liste d'étiquettes, mais on peut calculer chaque attribut indépendamment : une déduction locale ne porte pas seulement sur une étiquette entière, mais sur tout attribut. Par exemple, l'accord dans le syntagme nominal est propagé tant que possible.

3.2.3. Les déductions contextuelles dans le tagging : leur champ d'action théorique

Tentons de circonscrire a priori le champ d'action des déductions contextuelles.

Définissons un segment intermédiaire entre les mots et les phrases : le syntagme non récursif (SNR par la suite), ou syntagme simple, ou syntagme noyau, ou syntagme sans ses syntagmes compléments, ou "core phrase", ou "chunk" dans la littérature en anglais (voir Abney 1996). Ce segment est stable entre des langues différentes, et trouve approximativement son équivalent à l'oral sous la forme du groupe accentuel.

On a donc une hiérarchie de segments à trois niveaux : mots, SNR, phrases, hiérarchie où le segment d'un niveau est constitué de segments du niveau inférieur : un tout est d'un type différent du type de ses

parties, contrairement au syntagme récursif, qui est constitué de mots et de syntagmes récursifs (certaines parties sont de même type que le tout).

Les mots dans un SNR, et les SNR dans une phrase ont des comportements très différents : les mots d'un SNR forment un agrégat très contraint autour d'un nom ou d'un verbe, une structure stable et explicitable exhaustivement, mais les SNR dans une phrase sont soumis à des contraintes plus relâchées, et forment des structures instables, dont l'explicitation exhaustive nous semble actuellement un objectif hors de portée.

À titre d'illustration, dans "Le Bourgeois Gentilhomme", Molière fait permuter les SNR dans la phrase, mais laisse inchangé l'ordre des mots dans les SNR :

Belle marquise , vos beaux yeux me font mourir d'amour .
D'amour mourir me font , belle marquise , vos beaux yeux .
Vos beaux yeux d'amour me font , belle marquise , mourir .

On ne peut alors considérer une phrase comme une suite continue, indifférenciée de mots, dans laquelle toutes les contiguïtés seraient équivalentes. On doit considérer de manière différente une contiguïté de deux tokens à l'intérieur d'un SNR, et une contiguïté de deux tokens à la frontière de deux SNR contigus. Une déduction contextuelle sûre devra donc s'appuyer sur la structure interne stable du SNR et être interne au SNR, et non entre le dernier token d'un SNR et le premier du SNR suivant.

Comme un SNR est constitué d'un élément central lexical (nom ou verbe), entouré d'éléments périphériques grammaticaux (prépositions, déterminants, clitiques, adverbes, ...), la déduction contextuelle canonique consiste en une déduction de la forme :

étiquette d'un mot grammatical étiquette du mot lexical contigu

Plus précisément, un mot grammatical marque le plus souvent (en français) le début d'un SNR ainsi que le type de ce SNR, ce qui signifie que la déduction est indirecte car elle passe par le type du SNR :

étiquette d'un mot grammatical type du SNR étiquette du mot lexical contigu
ou : étiquette d'un mot grammatical type du SNR étiquette du mot gramm. contigu

Exemples :

une ferme	*une* > déterminant	*une ferme* SNR nominal	*ferme* > nom
ne ferme	*ne* > négation	*ne ferme* SNR verbal	*ferme* > verbe
je le	*je* > pronom sujet	*je le ...* SNR verbal	*le* > pronom objet
le bois	*le* > pronom objet	*... le bois* SNR verbal	*bois* > verbe

On peut alors définir un cas où il est impossible de faire une déduction locale et interne au SNR : c'est le cas d'un SNR constitué d'un seul mot lexical. Cette déduction locale impossible ne porte pas à conséquence si ce mot n'a qu'une seule catégorie possible.

Dans l'exemple suivant, ***remporte*** constitue à lui seul un SNR, et ne peut être que verbe :

*Comme prévu, M. Museveni **remporte** la quasi-totalité des votes dans l'Ouest, ...*

Mais, si ce mot a plusieurs catégories possibles, alors la décision ne peut pas être prise par déduction contextuelle locale : la décision ne pourra être prise que par la mise en relation du SNR de type multiple avec un SNR de type connu. Dans le corpus qui a servi à l'évaluation des essais de l'action GRACE (action internationale d'évaluation comparative des "taggers" du français - voir en 4.1) en septembre 1996 (environ 10 000 mots du journal Le Monde, fichier "lemon06"), environ 1% des mots sont dans ce dernier cas (1/6 sont des noms, 1/6 sont des verbes, et 2/3 sont des adjectifs ou des participes passés en position adjectivale).

Dans l'exemple suivant, ***montre*** constitue à lui seul un SNR, et peut être verbe ou nom ; la décision ne peut pas être prise localement, mais par la mise en relation avec son sujet ***présence*** :

*La **présence** de Florence Arthaud au milieu d'un plateau de spécialistes **montre** que cette Transat a été la course la plus disputée de ces dix dernières années.*

Cette déduction est du type :

présence sujet potentiel de *montre* si *montre* est un SNR verbal

montre SNR verbal *montre* verbe

En conclusion de cette section, l'étiquetage des mots qui forment à eux seuls un SNR de type multiple est impossible par déduction contextuelle et il devra donc être confié à une étape ultérieure de l'analyse. Notons en outre que tout étiquetage fondé sur une relation entre deux SNR est aussi impossible par déduction contextuelle ; citons par exemple : la résolution de *de d' du des* préposition ou partitif-article, la résolution de *que qu'* conjonction ou pronom relatif, les genre et nombre des pronoms relatifs selon leur antécédent, les genre, nombre, cas, type des clitiques réfléchis et des *nous* et *vous*, toute propagation de genre, nombre, personne par accord entre SNR (sujet - verbe, antécédent - pronom relatif, ...), en anglais, la résolution des formes verbales en *-ed*, prétérit ou participe passé.

3.2.4. Les ressources lexicales dans le tagging

Dans les sections précédentes, nous avons focalisé notre attention sur les déductions contextuelles. Mais ces déductions s'appuient sur des ressources lexicales, et doivent s'articuler avec elles. Étudions de quelle manière.

3.2.4.1. Trois articulations possibles entre ressources lexicales et déductions contextuelles

• Voyons comment le problème du tagging est habituellement posé : pour chaque token, toutes ses catégories sont exhaustivement énumérées à partir de sources d'informations lexicales (lexique de lemmes ou de formes, reconnaisseur de formes verbales, règles sur les finales - ou "guesser" - pour les mots absents du lexique), et le tagger doit choisir parmi les catégories possibles (comme l'homographie polycatégorielle est couramment appelée "ambiguïté", ce processus de choix est souvent appelé "désambiguïsation").

Une première solution pour effectuer un choix est d'attribuer au contexte le rôle de supprimer des catégories possibles de cette liste (c'est le cas des grammaires de contraintes - voir Voutilainen 1994). Nous appellerons une telle déduction : "déduction négative", qui correspond à une règle de la forme :

dans tel contexte, tel token **ne peut pas** avoir telles catégories

Le principal défaut d'une déduction négative est que la catégorie attendue du token doit appartenir à la liste des catégories possibles, ce qui revient à faire l'hypothèse que tout token appartient au lexique et est muni de la liste exhaustive de ses catégories possibles, hypothèse manifestement fausse pour les mots lexicaux, même avec un "guesser".

• Une autre solution pour choisir est d'utiliser des "déductions affirmatives", de la forme :

dans tel contexte, tel token **a** telle catégorie

Cette solution a l'avantage de pallier cette double incomplétude des ressources lexicales : certains tokens n'y sont pas, et certains tokens présents n'ont pas toutes leurs catégories possibles.

Par exemple, dans *je positive, positive* est pré-étiqueté adjectif dans le lexique, et la déduction est la suivante :

je positive *je* > pronom sujet *je positive* SNR verbal *positive* > verbe

• Mais il y a encore une autre manière de poser le problème : on remarque que les différentes catégories possibles d'un token sont loin d'être équiprobables : en général une catégorie est de beaucoup

la plus fréquente. Prenons l'exemple caricatural de *le l' la les* dans ce même corpus de 10 687 mots [9] du journal Le Monde : 1054 occurrences qui se répartissent en 1029 déterminants (97,6%), et 25 pronoms clitiques objet (2,4%).

Au lieu de poser au départ des déductions que ces graphies peuvent être déterminants ou pronoms, posons qu'elles sont déterminants **par défaut** (ces informations par défaut sont codées dans le lexique), et qu'elles seront pronoms dans un contexte particulier (ces informations liées au contexte sont codées dans les règles de déduction contextuelle du tagging) :

pronom atone sujet	suivi de *le l' la les*	*le l' la les* pronoms atones objet
négation *ne*	suivie de *le l' la les*	*le l' la les* pronoms atones objet
le l' la les	suivi d'un verbe sûr	*le l' la les* pronoms atones objet

On trouvera une démarche analogue dans (Chanod et Tapanainen 1995, page 151, 4.2.2), mais restreinte à certains mots grammaticaux qui ont une homographie très rare avec un mot lexical (*est, cela, avions*), homographie détectée à l'aide du contexte.

Donner une catégorie par défaut dans le lexique, et la modifier éventuellement par le contexte constitue en quelque sorte une implémentation du concept de translation de Lucien Tesnière (voir Tesnière 1959, à partir de la page 361). Un exemple généralisé est donné dans SYLEX, l'analyseur de Constant (voir Constant 1991).

3.2.4.2. Types d'homographies polycatégorielles, et étude statistique

Étudions les principaux types d'homographies, et faisons-en une étude statistique sur ce même corpus du journal Le Monde (de 10 687 mots), corpus pour lequel nous disposons d'un étiquetage manuel réalisé par des linguistes du comité de coordination de l'action GRACE, selon des spécifications proposées par le comité de coordination, puis discutées durant l'adjudication des essais. En consultant automatiquement des ressources lexicales incluant les homographies, nous obtenons les résultats suivants, en mettant en évidence l'alternative la plus rare :

- homographies sur les catégories de mots grammaticaux :

2,4% de pronoms	sur 1054 *le l' la les*	homographes déterminant / pronom
8,5% de partitifs-articles	sur 1088 *de d' du des*	homographes préposition / partitif-article
16,2% de pronoms relatifs	sur 130 *que qu'*	homographes conjonction / pronom relatif
4,7% de noms	sur 171 *est être avoir*	homographes verbe auxiliaire / nom
17,1% de noms	sur 111 *bien mal moins plus* ...	homographes adverbe / nom

- homographies sur les catégories de mots lexicaux :

16,4% de verbes	sur 487	homographes nom / verbe non auxiliaire
33,8% de noms	sur 714	homographes adjectif / nom

(ces derniers concernent les absents du lexique, normalement étiquetés par notre analyseur adjectif ou nom, par déduction locale dans le SNR nominal ou verbal)

- homographies sur les attributs de noms, adjectifs, pronoms, verbes :

24,6% de féminins	sur 2475	homographes masculin / féminin
25,7% de pluriels	sur 501	homographes singulier / pluriel
3,8% de subjonctifs	sur 185	homographes indicatif / subjonctif

Cette étude confirme qu'une des alternatives est toujours beaucoup plus fréquente.

[9] Ce comptage a été effectué automatiquement dans un éditeur (BBEdit sur Mac) : l'apostrophe et le tiret sont des séparateurs, la ponctuation et les guillemets ne sont pas comptés comme mots.

3.2.4.3. Une expérience d'évaluation de l'étiquetage par défaut

Pour évaluer l'intérêt et l'importance des étiquettes par défaut, codées dans les ressources lexicales, par rapport aux déductions contextuelles, faisons l'expérience d'étiqueter uniquement avec les étiquettes par défaut des ressources lexicales, sans **aucune** déduction contextuelle locale. Comme valeurs par défaut, prenons systématiquement l'alternative la plus fréquente :

- homographies sur les catégories de mots grammaticaux :
  - déterminant pour *le l' la les* homographes déterminant / pronom
  - préposition pour *de d' du des* homographes préposition / partitif-article
  - conjonction pour *que qu'* homographes conjonction / pronom relatif
  - verbe pour *est être avoir* homographes verbe auxiliaire / nom
  - adverbe pour les homographes adverbe / nom
- homographies sur les catégories de mots lexicaux :
  - nom pour les homographes nom / verbe non auxiliaire
  - adjectif pour les homographes adjectif / nom
- homographies sur les attributs de noms, adjectifs, pronoms, verbes :
  - masculin pour les homographes masculin / féminin
  - singulier pour les homographes singulier / pluriel
  - indicatif pour les homographes indicatif / subjonctif

L'expérience porte sur ce même corpus de 10 687 mots du journal Le Monde des essais de l'action GRACE, étiqueté à la main par des linguistes du comité GRACE. Nous pouvons alors comparer un étiquetage automatique avec l'étiquetage manuel [10].

Le jeu d'étiquettes GRACE est dérivé du jeu MULTEXT [11] ; il comprend 11 catégories de base, avec 0 à 6 attributs ayant de 2 à 8 valeurs, ce qui donne 311 étiquettes différentes. La tokenisation GRACE est très fine : tout mot contenant une apostrophe ou un tiret est subdivisé, et apostrophes et tirets sont des tokens (respectivement 4,8% et 1,0% des tokens), comme le reste de la ponctuation (10,5% des tokens), ce qui donne 12741 "tokens" pour 10 687 "mots".

Dans ces conditions expérimentales, et en reproduisant aussi fidèlement que possible le protocole d'évaluation GRACE, on obtient alors les résultats suivants :

aucune étiquette multiple (décision = 1,0000)

92,1% des tokens ont la même catégorie de base que dans le corpus de référence
(précision sur les catégories de base = 0,9208)

82,5% des tokens ont exactement la même étiquette que dans le corpus de référence
(précision sur les étiquettes complètes = 0,8254)

En conclusion de notre expérience, nous constatons que les ressources lexicales avec des **valeurs par défaut**, sans aucune déduction contextuelle, permettent d'obtenir 9 catégories de base sur 10 égales à celles du corpus de référence, sur le jeu de 11 catégories de base, et 8 étiquettes complètes sur 10 égales à celles du corpus de référence, sur le jeu de 311 étiquettes complètes (notons qu'il est important de préciser avec quelle tokenisation et avec quel jeu d'étiquettes on obtient telle précision). Avec les seules ressources lexicales, l'étiquetage automatique a ainsi parcouru la majeure partie du chemin vers l'étiquetage manuel.

---

[10] Pour les mettre en conformité avec les spécifications de l'étiquetage GRACE résultant de l'adjudication des essais, nous avons modifié environ 2% des tokens du corpus de référence étiqueté à la main.

[11] http://www.lpl.univ-aix.fr/projects/multext

Nous faisons l'hypothèse que la valeur par défaut (venant du lexique), jusqu'à preuve du contraire (apportée par le contexte), est une propriété générale des langues, permettant une économie dans les processus de communication langagiers.

3.2.4.4. Contributions des déductions contextuelles et des mises en relation

En introduisant successivement les déductions contextuelles, puis les mises en relations des SNR, voici un tableau résumant les valeurs obtenues en précision (la décision reste égale à 1,0000) :

ressources		précision sur les 11 catégories de base		précision sur les 311 étiquettes complètes	
lexique avec valeurs par défaut		92,1%	+92,1%	82,5%	**+82,5%**
+ déductions contextuelles :	absolu	98,0%		94,8%	
	relatif		+5,9%		**+12,3%**
+ mises en relation :	absolu	99,3%		97,1%	
	relatif		+1,3%		**+2,3%**

En comparant les contributions des trois types de ressources, on observe la répartition suivante (en précision sur les étiquettes complètes) :

- le lexique avec des valeurs par défaut résout 82,5% des tokens,
- la déduction contextuelle améliore la précision pour 12,3% des tokens, ce qui fait plafonner la précision à 94,8%,
- et enfin la mise en relation des SNR permet de résoudre 2,3% des tokens, qui sont environ pour moitié les mots qui forment à eux seuls un SNR de type multiple (ceux évoqués ci-dessus en section 2), et pour moitié certains mots grammaticaux étiquetés par la mise en relation : résolution de *de d' du des* préposition ou partitif-article, résolution de *que qu'* conjonction ou pronom relatif, genre et nombre des pronoms relatifs selon leur antécédent, genre, nombre, cas, type des clitiques réfléchis et des *nous* et *vous*.

L'écart de 3% en précision sur les étiquettes complètes qui sépare les 97,1% des 100% est dû pour moitié à des erreurs de notre analyseur, et pour moitié à un écart incompressible dû aux faits que l'étiquetage manuel ne peut être d'une régularité parfaite et que les spécifications aussi précises soient-elles ne peuvent pas prévoir tous les cas réels.

Remarque sur les valeurs des précisions données ci-dessus : ces valeurs ne peuvent pas être comparées avec les valeurs obtenues par d'autres taggers dans d'autres conditions expérimentales : tokenisation, jeu d'étiquettes (nombre d'étiquettes, jeu de type traditionnel ou distributionnel), corpus, spécifications d'étiquetage du corpus de référence, protocole et métrique de calcul des écarts entre résultats calculés et corpus de référence. L'intérêt de ces valeurs est surtout de permettre ici la comparaison relative entre les apports des trois types de ressources dans la précision du tagging. La comparaison des performances de taggers différents ne peut se faire avec rigueur que s'ils sont comparés dans le cadre de conditions expérimentales strictement identiques, comme dans l'action GRACE par exemple. Remarquons l'importance du nombre d'étiquettes dans la comparaison des précisions de différents taggers : un petit nombre d'étiquettes rend la décision plus facile (la probabilité est plus grande de tomber par hasard sur la bonne étiquette), mais jusqu'à un certain point, car un trop petit nombre d'étiquettes rendrait les déductions contextuelles locales plus difficiles par manque de finesse et de régularité dans la description des contextes.

3.2.5. Définition du "token", définition du "tagset"

Le mot, qui exprime une segmentation conventionnelle et instable entre des langues différentes, et évolutive à l'intérieur d'une langue, n'est pas le bon candidat pour être systématiquement le "token", segment de phrase, unité traitée dans un traitement de langue : un mot peut être divisé en plusieurs tokens (cas des amalgames, des mots composés), ou plusieurs mots peuvent être regroupés en un seul token : locutions diverses, noms propres composés, numéraux composés, mots composés. Le token est défini en rapport avec l'objectif du traitement.

Le "tagset", ou jeu d'étiquettes des tokens, doit-il, par fidélité aux traditions, reprendre les parties du discours? Rappelons-nous que les parties du discours constituent une taxinomie traditionnelle et empirique des "mots", consacrée surtout à l'enseignement, une théorie parmi de nombreuses autres possibles (voir un exemple de théorie originale dans Tesnière 1959, pages 51 à 94), et que nous avons toute liberté d'en créer une autre, adéquate au tagging. Posons-nous alors la question : quelles seraient les caractéristiques d'un jeu d'étiquettes adéquat aux déductions contextuelles, déductions sur les régularités des contiguïtés des catégories, sur les **régularités distributionnelles**, captées à travers le filtre du jeu d'étiquettes.

Par exemple, ces trois étiquettes traditionnelles caractérisent un SNR nominal :

"une" : article "cette" : adjectif démonstratif "sa" : adjectif possessif

Elles sont maintenant plus souvent regroupées sous une étiquette unique : le déterminant, étiquette fondée sur la régularité de la classe distributionnelle des déterminants, différenciée de celle des adjectifs.

En revanche, "il", "elle", "nous", "y", "ceux", "qui", "dont" sont tous des pronoms, mais ils ont des distributions bien différentes !

Donc, pour capter (puis utiliser) des régularités distributionnelles, chaque étiquette du jeu d'étiquettes doit définir une classe distributionnelle de tokens. Par exemple, parmi les adjectifs, on devra différencier les épithètes antéposées, les épithètes postposées, et les attributs, car leurs distributions sont différentes.

Un jeu d'étiquettes distributionnel, associé au concept de SNR nominal ou verbal, implique que ce jeu est partitionné en deux sous-ensembles (leur intersection est vide) : le jeu des étiquettes dans le SNR nominal et le jeu des étiquettes dans le SNR verbal. Par exemple, l'étiquette de l'adjectif épithète, dans le SNR nominal, est différente de celle de l'adjectif attribut, dans le SNR verbal [12].

3.2.6. Les concepts du tagging : un renouveau pour l'analyse syntaxique

Comme nous l'avons vu en section 3.2.1.2., l'apport conceptuel principal et original du tagging est l'explicitation des processus, associé à l'abandon de l'explicitation des structures. Les processus explicités sont des déductions contextuelles.

Nous y ajoutons le concept central de syntagme non récursif (défini en section 3.2.2), des ressources lexicales avec des valeurs par défaut (étudiées en section 3.2.4) généralisées pour l'ensemble des mots grammaticaux, et un jeu d'étiquettes distributionnel (présenté en section 3.2.5). Le tagger, devenu moteur de déduction contextuelle, produit en sortie, non seulement une liste de tokens étiquetés, mais aussi une liste de syntagmes non récursifs délimités et typés. Environ 1% des tokens ont à la fois les deux propriétés suivantes : ils ont plusieurs catégories possibles parmi verbe, nom, adjectif et ils constituent seuls un SNR de type multiple (voir ci-dessus en section 3.2.3). Les autres tokens et les autres SNR ont respectivement une catégorie unique et un type unique. Après les déductions contextuelles, les catégories par défaut de *de d' du des* et *que qu'* sont encore respectivement préposition et conjonction.

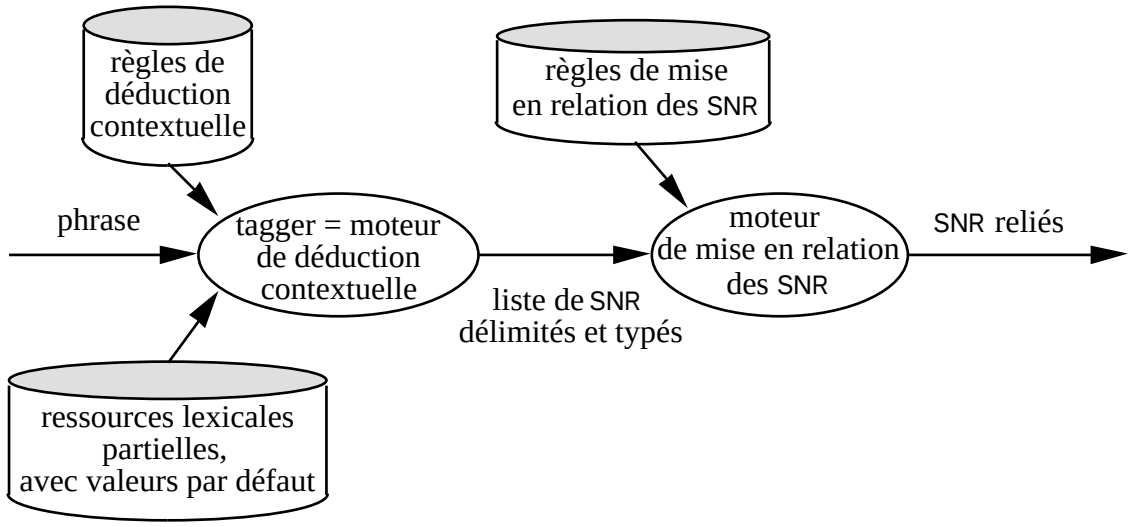
[12] Notre tagset est disponible sur : http://www.info.unicaen.fr/~giguet/java/textes/cat_mots_SNR.html

Que manque-t-il alors pour terminer l'analyse syntaxique? Il faut relier les SNR (relations de dépendances, d'apposition, de coordination, d'antécédence, ...).

Dans l'esprit du tagging (l'explicitation des processus), nous avons conçu un processus de mise en relation des SNR, qui ne fait aucune hypothèse explicite sur les structures syntaxiques situées entre les SNR reliés, ni sur la distance qui les sépare. Ce processus est implémenté comme dans le tagger, sous la forme d'un moteur qui interprète des règles [13] qui explicitent le processus. Ses principes et son implémentation sont décrits dans (Giguet et Vergne 1997) et ci-dessous en 3.3.

L'ensemble, les ressources lexicales, les deux moteurs et les deux bases de règles, constituent un analyseur de complexité linéaire, dans lequel les processus sont explicités et qui produit en sortie les structures syntaxiques sous la forme de syntagmes non récursifs reliés (voir figure 4 ci-dessous).

Figure 4 : Tagger intégré dans notre analyseur



Les résultats sur des corpus variés en français, articles de journaux (Le Monde), littérature, textes scientifiques sont visibles sur internet à l'adresse : http://www.info.unicaen.fr/~giguet. C'est cet analyseur que nous avons utilisé pour réaliser l'expérience rapportée ci-dessus (en section 3.2.4.3.). C'est aussi cet analyseur qui nous a permis d'obtenir la première place à l'action GRACE (voir ci-dessous en 4.1), en lui ajoutant en module final une fonction de transfert qui transforme notre tokenisation et notre étiquetage en la tokenisation et l'étiquetage GRACE. Les déductions ayant été faites sur notre jeu d'étiquettes, ce transfert permet de faire les évaluations sur le jeu d'étiquettes GRACE. Ce dernier, à cause de sa trop grande fidélité aux parties du discours traditionnelles et de ses propriétés distributionnelles insuffisantes, n'est pas très approprié aux déductions contextuelles. Mais il a l'intérêt majeur d'être le résultat d'un large consensus de la communauté scientifique du tagging du français, et de permettre de faire les calculs d'écart entre le corpus de référence et les résultats de différents taggers dans des conditions expérimentales unifiées et rigoureuses.

3.2.7. Conclusion

Face aux difficultés rencontrées par l'analyse syntaxique traditionnelle, principalement dues à son aspect combinatoire, et à l'obligation de disposer d'un inventaire exhaustif des structures syntaxiques d'une langue, le tagging constitue une échappatoire prometteuse, mais il s'est surtout centré sur ses aspects opératoires, et s'est peu interrogé sur ses bases théoriques.

[13] Dans les deux cas, les règles sont de la forme : conditions actions.

Nous avons tenté de pallier cette lacune, en montrant que le tagging prend le contre-pied de l'analyse syntaxique traditionnelle en mettant l'accent sur l'explicitation des processus, au lieu d'attendre des structures explicitées dans des grammaires formelles, et que, par là même, il ouvre la voie au renouveau de l'analyse syntaxique en la fondant sur l'explicitation des processus : processus de déduction contextuelle dans les syntagmes non récursifs, et processus de mise en relation des syntagmes non récursifs. On étend ainsi à toute l'analyse syntaxique les propriétés calculatoires du tagging, et on obtient ainsi des algorithmes de complexité linéaire en temps.

3.3. L'analyseur 98 : analyse non combinatoire de phrases

3.3.1. Dix années d'évolution

Cet analyseur est celui qui a remporté la première place à l'action GRACE (voir ci-dessous en 4.1.).

Ses sources sont en évolution depuis 10 ans, ce qui en fait une "usine à gaz" qui reste opératoire, mais dont le développement est abandonné depuis fin 1998 (il a fallu l'abandonner en emportant les concepts).

Dès le départ, depuis la thèse, à cause de ma formation initiale autodidacte en linguistique et en informatique, l'analyseur s'est situé en dehors des paradigmes classiques de l'analyse syntaxique automatique, faisant ainsi l'économie de leur abandon, en l'occurrence : les bases linguistiques chomskiennes, la modélisation des structures syntaxiques (dont le syntagme récursif) par des grammaires formelles, l'opposition apparente entre les modèles de Chomsky et de Tesnière, l'analyse conçue sur le modèle de la compilation, comme la reconnaissance de structures attendues et explicitées exhaustivement, avec des ressources lexicales supposées exhaustives, l'hypothèse qu'une langue peut être traitée telle un langage de programmation, clos, fini, exhaustivement explicité (cf. ci-dessus en 3.1.).

Dès le départ, l'analyseur a eu les caractéristiques suivantes : hypothèse qu'une langue est ouverte, et donc que les ressources sont nécessairement incomplètes, pas d'attente de structure de phrase a priori, hiérarchie fixe de segments non récursifs, mise au point par travail expérimental sur corpus, mise en relation non pas seulement des mots, comme dans les grammaires de dépendance classiques, mais aussi des SNR.

En 10 ans, les évolutions fondamentales furent :

- l'abandon du processus combinatoire (avril 93), précédée par une utilisation de plus en plus massive du contexte, qui a permis d'abord de réduire la combinatoire, puis de l'abandonner (cf. ci-dessus en 3.0); la seule cause de combinatoire était l'homographie polycatégorielle; c'était le début de l'abandon progressif de la reconnaissance de structures pour le calcul de structures;

- la démarche multilingue associée à la déclaration des ressources linguistiques : travail sur l'espagnol (à partir de janvier 93) et sur l'anglais (à partir de juillet 94); déclaration de toute ressource spécifique à une langue sous forme de ressources monolingues : ressources lexicales + règles interprétées (à partir de février 94);

- les valeurs par défaut, venant des ressources lexicales, modifiées ensuite éventuellement par le calcul contrôlé par l'application des règles (à partir du printemps 94), comme outil radical pour abandonner le paradigme du choix pour celui du calcul;

- l'algorithme de mise en relation des SNR (Noël 94).

À partir de début 99, un nouvel analyseur est en cours de développement, pour intégrer proprement tous les nouveaux concepts élaborés (voir ci-dessous en 3.4.).

À quoi a servi cet analyseur ? C'est prioritairement un "logiciel d'étude" (terme dû à Jacques Courcil et Anne Nicolle), qui à servi à élaborer et valider des concepts de syntaxe, et des concepts d'analyse syntaxique. Notre expérience des collaborations industrielles nous montre clairement qu'il est important de bien dissocier logiciel d'étude et logiciel d'application, pour ne pas courir deux lièvres à la fois : l'élaboration et la validation de concepts d'une part, la réalisation d'une tâche effective en situation industrielle d'autre part. Une fois les concepts élaborés, puis validés au moyen d'un logiciel d'étude, il faut transposer ces concepts dans un logiciel d'application avec liberté de choix de la nouvelle implémentation : machine, système, langage, environnement de développement, en fonction de l'application. Par exemple, l'analyseur a été transposé (en C^{++}) par Gérald Vannier en moins de 2 mois dans la synthèse vocale logicielle KALI.

La transposition est d'autant plus facile que les ressources sont déclaratives, car la transposition se limite à la réécriture des moteurs, avec d'autres priorités : vitesse d'exécution, facilité de maintenance et éventuellement portabilité.

Au sujet du langage de programmation, la question de son choix se pose différemment pour un logiciel d'étude ou pour une application. Un logiciel d'étude étant un système servant à explorer des solutions nouvelles, on a besoin d'un outil très général permettant d'implémenter des algorithmes très variés. Un langage de programmation est un outil d'expression de solution, qui facilite certains types d'expression, et qui en rend d'autres plus difficiles ou impossibles, exactement comme une langue. Lisp et Prolog, par exemple, sont tous deux des langages à (trop) forte personnalité, c'est-à-dire très pratiques pour résoudre une classe restreinte de types de problèmes par une classe restreinte de solutions, mais difficilement maniables et contraignants en dehors de leur zone privilégiée. C'est pour ces raisons que je développe en un langage généraliste permettant de programmer en procédural, en fonctionnel, en récursif, en arborescent, ..., au choix. Cette liberté de choix des solutions se paye de plus de lourdeur. Donc depuis le début, je développe en Pascal : UCSD sur AppleII (seul langage structuré sur micro à cette époque), puis TurboPascal sur Mac, puis Think Pascal sur Mac.

3.3.2. Architecture et algorithme général

Cette architecture est à considérer par contraste avec celle de l'analyseur 99 décrit ci-dessous en 3.4. En voici les caractéristiques principales :

• C'est toujours un analyseur de phrase : le tokeniseur a la double fonction de découper le flux de caractères entrant en tokens, et d'arrêter ce découpage à une fin de phrase, pour fournir cette liste de tokens de l'unique phrase courante au moteur de déduction contextuelle; et le traitement est répétitif par phrase.

• L'analyseur est constitué de 4 modules exécutés en chaîne de traitement pour une phrase.

• Ces 4 modules constituent 2 couches successives (1 et 2) sur une phrase, soit à 2 reprises :

  a) changer de représentation (et d'échelle) = regrouper des éléments

  b) calculer sur la représentation = passer des règles sur la représentation.

• D'où l'algorithme général :

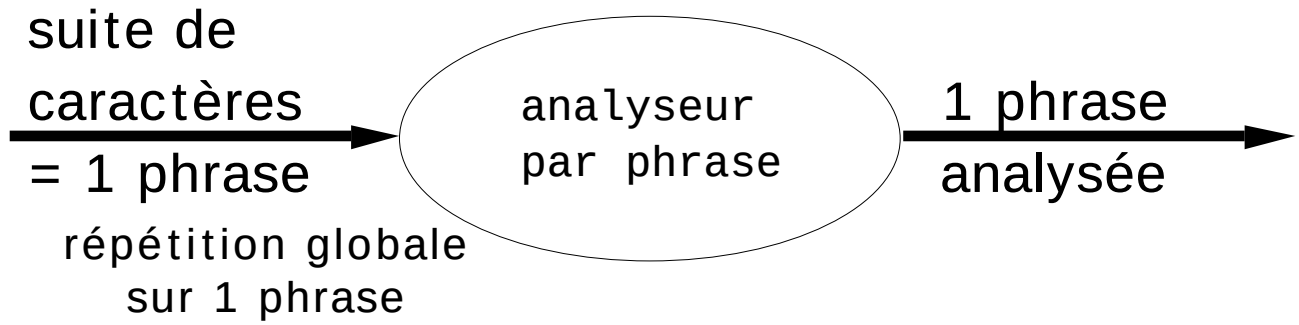
```
analyseur 98 :
   répéter        (par phrase)
   |  1a. tokeniser le flux de caractères = acquérir les tokens d'une phrase
   |  1b. passer les règles de déduction contextuelle sur les tokens de la phrase
   |  2a. calculer la représentation de la phrase en SNR
   |  2b. passer les règles de mise en relation des SNR de la phrase
   jusqu'à  la fin du texte
```

Le module b) de chaque couche est un moteur qui applique des règles sur des éléments :

```
    b. passer des règles sur des éléments de la phrase courante :
         pour chaque élément de la phrase
           pour chaque règle
             si  la condition de la règle s'applique à l'élément et ses voisins
               alors  appliquer l'action de la règle à l'élément et ses voisins
```

Voici en figure 5 le schéma global de l'architecture de l'analyseur 98 :

Figure 5 : Schéma global de l'architecture de l'analyseur 98



La figure 6, page suivante, montre l'architecture de l'analyseur 98, constitué de modules de traitements en chaîne, à comparer avec l'analyseur 99, où les traitements ne sont plus en chaîne (cf. 3.4.).

3.3.3. Ressources lexicales

1) partition des ressources lexicales

Le problème à résoudre est d'attribuer au début du calcul une catégorie de départ à tout token entrant, alors que les ressources lexicales sont finies, mais qu'il y a une infinité de tokens possibles, et qu'un token connu peut avoir un emploi imprévu. Il est impossible d'énumérer exhaustivement tous les tokens possibles et tous leurs emplois possibles.

Une solution est d'avoir des ressources lexicales (forcément) partielles, dont les lacunes seront comblées par la suite du processus : les déductions contextuelles sur les tokens dans les SNR (pour presque tous les tokens), et (pour environ 2 à 3% de tokens) la mise en relation des SNR.

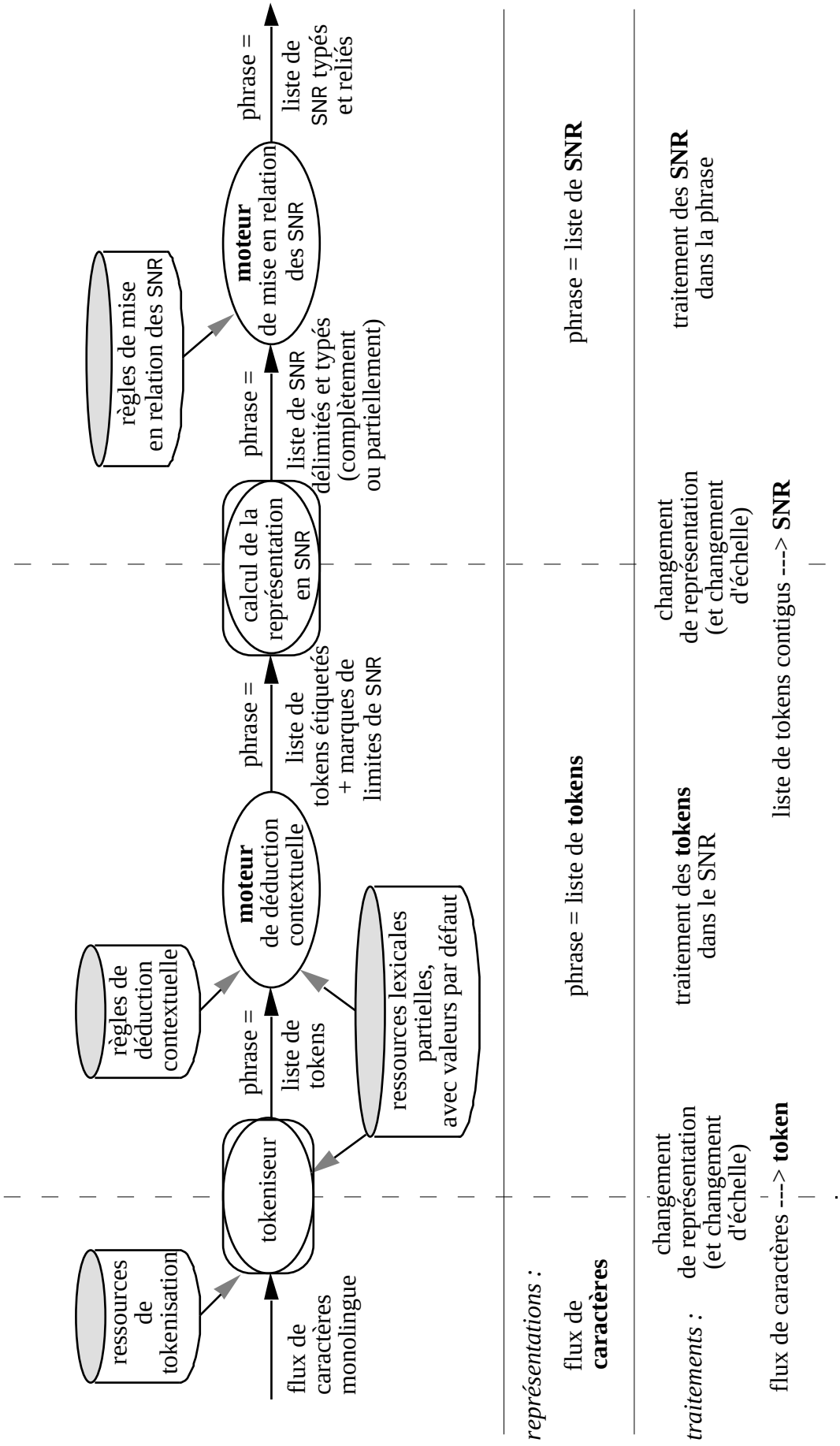
Pour pallier l'infinité des tokens possibles, on peut partitionner ainsi l'ensemble des tokens :

1- les mots grammaticaux, courts, énumérables exhaustivement, en faible nombre : conjonctions, prépositions, déterminants, pronoms, négations, ponctuation

2- adverbes non dérivés d'adjectif

3- adverbes dérivés d'adjectif (en -ment)

Figure 6 : Architecture de l'analyseur 98



4- adjectifs numéraux cardinaux en lettres

5- adjectifs numéraux cardinaux en chiffres arabes ou romains

6- adjectifs numéraux ordinaux

7- adjectifs couramment antéposables, dont les indéfinis

8- noms propres repérés par leur initiale majuscule

9- formes verbales supposées énumérables exhaustivement

10- tout le reste, qui ne peut plus être que nom ou adjectif, et qui est le lieu principal de la néologie.

Cette partition est à la base de la conception des ressources lexicales et de leur ordre d'application.

Remarque : mots grammaticaux et finales sont des morphèmes, les premiers détachés avant le mot lexical (nom, verbe, adjectif, adverbe), les seconds attachés après (cas du français); cette caractéristique d'être détaché ou attaché est contingente à la segmentation actuelle en mots; ils ont des propriétés analogues : courts, énumérables exhaustivement, en faible nombre (voir Déjean 98).

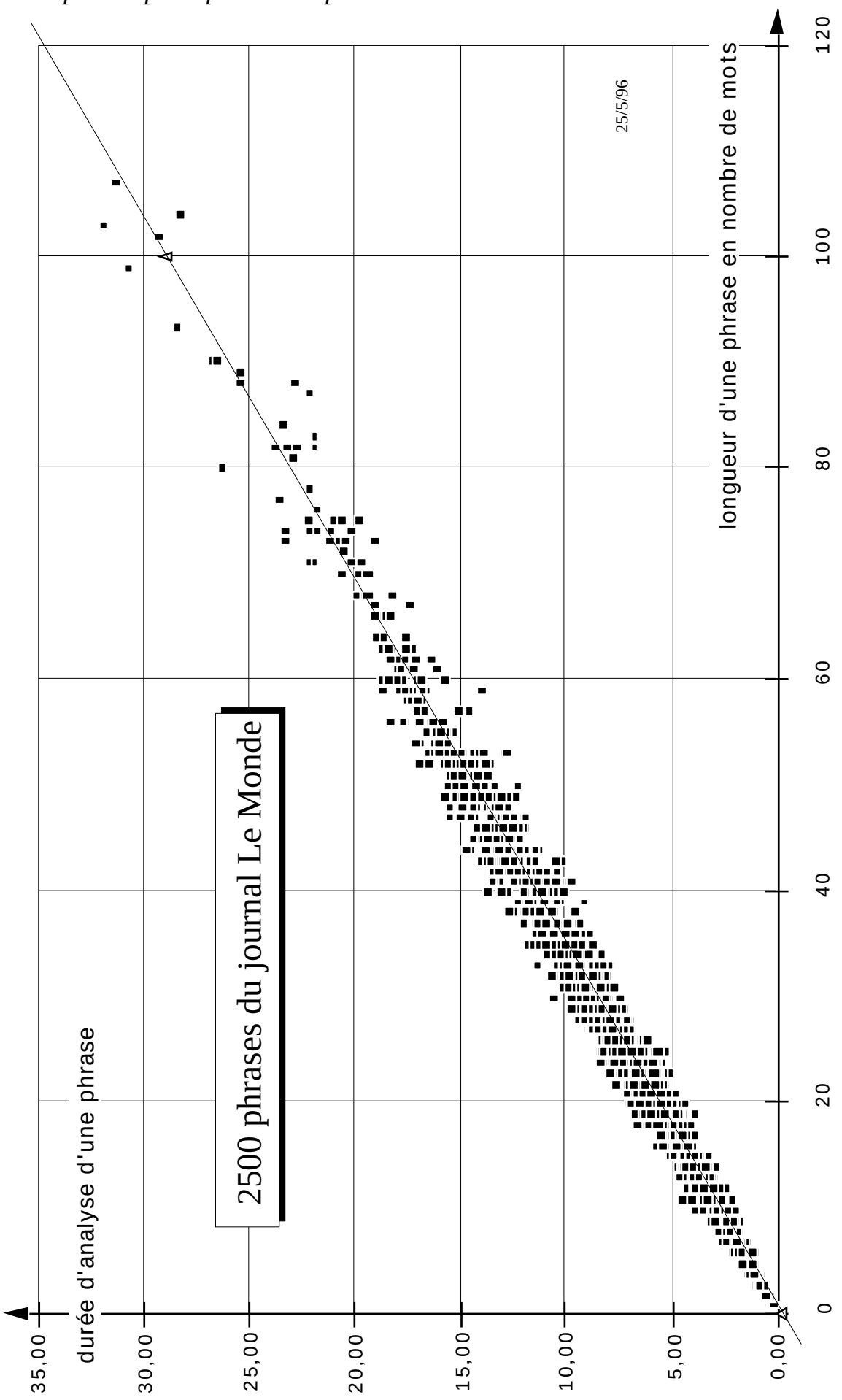
Pour pallier l'impossibilité d'énumérer toutes les catégories possibles d'un token, les ressources lexicales décrivent des valeurs par défaut (1 catégorie pour les mots grammaticaux, plusieurs pour les formes verbales homographes polycatégorielles), qui peuvent être modifiées par déduction contextuelle: par intersection de l'ensemble initial des catégories possibles avec un ensemble calculé d'après le contexte, ou par affectation d'une nouvelle catégorie.

2) chronologie d'exploitation des ressources lexicales

Les ressources lexicales intègrent les sous-ensembles de la partition ci-dessus et sont subdivisées en 5 parties exploitées dans cet ordre (effectifs pour le français) :

lexique.<langue>	lexique des mots grammaticaux (400) avec 1 catégorie par défaut, + adverbes non dérivés d'adjectif (250) + adjectifs couramment antéposables (650) + noms sûrs (10 000) dont les attributs ne seraient pas calculables par les règles sur les finales (adjectifs et noms pour la résolution de nom-adjectif ou adjectif-nom) (sous-ensembles 1, 2, 4, 6, 7 de la partition)
numéraux cardinaux	règles : numéral en chiffres (arabes, romains) => catégorie, attributs (sous-ensemble 5 de la partition)
noms propres	règle : initiale majuscule => nom propre (sous-ensemble 8 de la partition)
verbes.<langue>	ressources de reconnaissance des formes verbales avec polycatégorie explicite (intraverbale, nominale, adjectivale), ce qui permet de traiter certains adjectifs et noms (sous-ensemble 9 de la partition)
finales.<langue>	règles sur les finales (1000) : finale => catégorie, attributs (sous-ensembles 3, 10 de la partition)

3.3.4. Complexité pratique sur corpus



3.3.5. Règles déclaratives et moteur

Voici les caractéristiques particulières des deux moteurs de l'analyseur 98 :

1) moteur de catégorisation des tokens et de délimitation des SNR

fonctions : marquer les limites des SNR et typer les SNR sur les tokens
compléter l'étiquetage des tokens d'après les types de SNR
(le type du SNR est un attribut du token)

structure de données concernée : la phrase représentée en une liste de tokens

éléments entrants : tokens

éléments sortants : tokens (la structure en SNR n'est pas encore calculée)

empan des règles : 1 à 5 tokens

nombre de règles pour le français : environ 300

opérateurs des conditions sur un token : graphie appartenant à un ensemble de graphies donné, catégorie appartenant à un ensemble de catégories donné (avec contrainte éventuelle sur un attribut), type du SNR du token, accord possible avec le token précédent, avec les opérateurs booléens **non et**

opérateurs des actions sur un token : catégoriser un token, marquer une limite de SNR ou un type de SNR sur un token, propager l'accord dans le SNR courant

éléments dont les attributs sont modifiés : tokens

Les conditions et les actions de toutes les règles de ce moteur portent sur des tokens contigus.

2) moteur de mise en relation des SNR dans la phrase

fonctions : relier les SNR dans la phrase (dépendances, coordinations, antécédances)
compléter l'étiquetage des SNR d'après les relations entre SNR
compléter l'étiquetage des tokens d'après les types de SNR

structures de données concernées : la phrase représentée en une liste de tokens et en une liste de SNR

éléments entrants : tokens et SNR

éléments sortants : tokens et SNR

empan des règles : 1 à 3 SNR

nombre de règles pour le français : environ 200

opérateurs des conditions sur un SNR : type du SNR courant, contiguïté avec un SNR de tel type, présence d'un SNR de tel type dans telle mémoire, accord possible du SNR courant avec un SNR dans telle mémoire, relation de dépendance du SNR courant avec un SNR de tel type, isomorphisme de coordination du SNR courant avec un SNR de telle mémoire, désignation du SNR à mettre en mémoire ou du SNR à relier (différent du SNR courant), avec les opérateurs booléens **non et**

opérateurs des actions sur un SNR : placer un SNR dans telle mémoire, relier un SNR (dépendance ou coordination) avec un SNR placé dans telle mémoire, typer un SNR, oublier un SNR dans une mémoire, catégoriser un token

éléments dont les attributs sont modifiés : tokens et SNR

Les règles de ce moteur implémentent exactement le processus décrit en 2.3.2.

Un processus de mise en relation est fondé sur 2 règles et 1 mémoire : une règle de mise en mémoire d'un SNR, et une règle de mise en relation d'un SNR désigné avec un SNR en mémoire.

3.3.6. Interaction entre les processus de mise en relation

L'interaction entre 2 processus de mise en relation est fondée sur le principe suivant : toute règle sur une mémoire A peut avoir une action sur une autre mémoire B : mémoriser ou oublier un SNR dans B.

Nous allons donner sur un exemple une trace simplifiée pour faire comprendre comment plusieurs processus de mise en relation interagissent par l'intermédiaire des mémoires :

```
   L'usine d'Eloyes dans les Vosges représente un investissement de 3,7 milliards
de yens.
```

En sortie du moteur de déduction contextuelle, les SNR sont délimités et la phrase est représentée en termes de SNR, représentation sur laquelle les mises en relation sont calculées :

```
 N {L' usine}                    N = SNR Nominal
pN {d' Eloyes}                   p = préposition
pN {dans les Vosges}
 V <représente>                  V = SNR Verbal
 N {un investissement}
pN {de 3,7 milliards}
pN {de yens}
```

Dans la trace suivante, l'axe syntagmatique est horizontal et orienté de gauche à droite, et l'axe des mémoires est vertical (on ne donne que 3 des 13 mémoires).

Chaque fois qu'une règle est appliquée, son action est commentée.

Il n'y a que 2 types de règles : celles qui placent le SNR courant dans une mémoire, et celles qui mettent le SNR courant en relation avec un SNR placé dans une mémoire.

```
état initial :              conventions et commentaires

   |                    position du front de règles sur l'axe syntagmatique (SNR courant)
   NpNpNVNpNpN          représentation de la phrase en SNR
   0 0 000 0 0          indice du régissant de chaque SNR
   12345678901          indices des SNR sur l'axe syntagmatique
π  ..........           état de la mémoire des sujets en attente de verbe
ß  ..........           état de la mémoire des N en attente de complément
ç  ..........           état de la mémoire des N en attente de coordonné
   NpNpNVNpNpN          (. = vide, * = oublié par attente terminée, - = oublié entre 2 SNR reliés)
              <         structure de donnée modifiée par une action d'une règle
_____________________________________________________________________________
   |                    front de règles en 1 (SNR courant = SNR 1)
   NpNpNVNpNpN
   0 0 000 0 0
   12345678901
π  N.......... <        1 N {L' usine} placé en attente de verbe
ß  ..........
ç  ..........
   NpNpNVNpNpN
   -----------
   NpNpNVNpNpN
   0 0 000 0 0
   12345678901
π  N..........
ß  N.......... <        1 N {L' usine} placé en attente de complément
ç  ..........
   NpNpNVNpNpN
```

```
   ----------
   NpNpNVNpNpN
   0 0 000 0 0
   12345678901
π  N.........
ß  N.........
ç  N......... <     1 N {L' usine} placé en attente de coordonné
   NpNpNVNpNpN
```

```
     |              front de règles en 3 (SNR courant = SNR 3)
   NpNpNVNpNpN
   0 0 000 0 0
   12345678901
π  N.........
ß  N.N........ <     3 pN {d'Eloyes} placé en attente de complément
ç  N.........
   NpNpNVNpNpN
   ----------
   NpNpNVNpNpN
   0 1 000 0 0 <     1 N {L' usine} <-- 3 pN {d'Eloyes} reliés
   12345678901
π  N.........
ß  N.N........
ç  N.........
   NpNpNVNpNpN
```

```
       |            front de règles en 5 (SNR courant = SNR 5)
   NpNpNVNpNpN
   0 1 000 0 0
   12345678901
π  N.........
ß  N.N.N...... <     5 pN {dans les Vosges} placé en attente de complément
ç  N.........
   NpNpNVNpNpN
   ----------
   NpNpNVNpNpN
   0 1 300 0 0 <     3 pN {d'Eloyes} <-- 5 pN {dans les Vosges} reliés
   12345678901
π  N.........
ß  N.N.N......
ç  N.........
   NpNpNVNpNpN
```

```
       |            front de règles en 6 (SNR courant = SNR 6)
   NpNpNVNpNpN
   0 1 310 0 0 <     1 N {L' usine} <-- 6 V <représente>  /fs3   reliés
   12345678901
π  *---------- <     * le sujet n'attend plus de verbe (oubli)
ß  N---------- <     N le sujet attend encore un éventuel complément après le verbe
ç  *---------- <     * le sujet n'attend plus de coordonné (oubli)
   NpNpNVNpNpN
```

`----` les compléments du sujet n'attendent plus aucun complément ni coordonné (oubli)

Cette dernière mise en relation entre sujet et verbe, par l'intermédiaire de la mémoire des sujets en attente de verbe (π) montre comment elle interagit avec d'autres mises en relation, en supprimant des attentes en cours dans des mémoires (la même ou d'autres) sur et entre les SNR reliés.

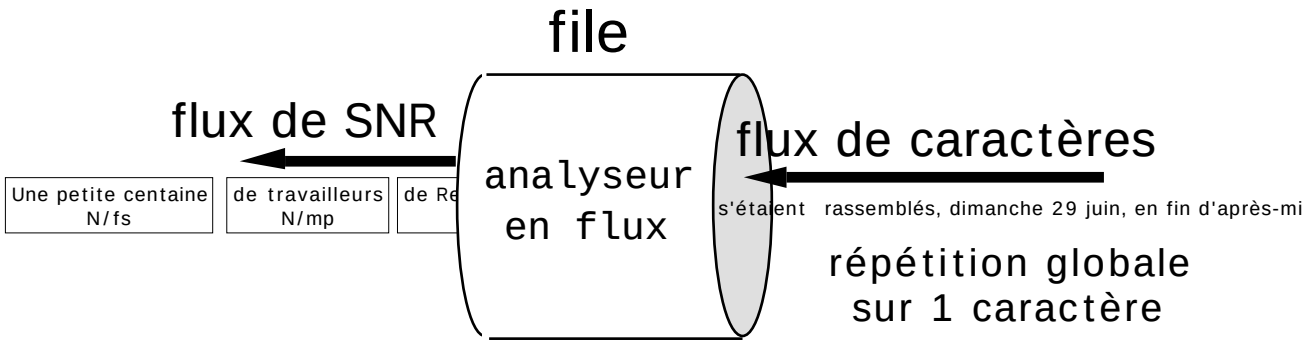
3.4. L'analyseur 99 : analyse non combinatoire de flux

Au moment où nous écrivons (juin 99), l'analyseur 99 est en cours de développement et sort un flux de syntagmes non récursifs (SNR) typés, constitués de tokens catégorisés, et reliés dans la proposition (sur le français et sur l'anglais). Pour vérifier qu'une même qualité que l'analyseur 98 est atteinte, il faut que les nouvelles bases de règles soient complétées et évaluées comparativement.

L'analyseur 99 intègre (et généralise) les nouveaux concepts élaborés précédemment avec une implémentation plus épurée :

- on n'a plus un traitement de phrase, mais un traitement de flux; tous les niveaux linguistiques (caractères, tokens, SNR, propositions, phrases, ...) sont traités simultanément en flux et non plus successivement pour une phrase :

Figure 7 : Flux d'entrée et flux de sortie de l'analyseur 99



- la déclaration des ressources propres à une langue est complète <=> les moteurs sont complètement indépendants de la langue traitée;

- l'analyseur est exclusivement constitué de moteurs qui traitent un flux d'éléments avec des règles déclaratives : chaque moteur est spécialisé dans le traitement d'un niveau linguistique;

- articulation entre moteurs : une action d'une règle au niveau n peut déclencher le moteur (et les règles) du niveau n+1, qui opère sur les éléments du niveau n+1;

- chaque niveau linguistique a sa structure de données propre, qui interagit avec les autres niveaux par les règles; par exemple, les tokens typent les SNR, et les SNR contraignent la catégorie des tokens, par affectation de valeurs d'attributs dans les structures respectives;

- le niveau de la proposition est explicitement représenté dans les structures de données;

- le principe des valeurs par défaut est généralisé : pour tous les tokens (mots grammaticaux et lexicaux, y compris les formes verbales), pour les SNR aussi; on généralise ainsi l'option : "l'ordinateur comme machine à calculer et non plus comme machine à choisir" (généraliser les valeurs par défaut revient à généraliser l'utilisation de l'ordinateur comme machine à calculer à partir de valeurs initiales par défaut au lieu de machine à choisir parmi des valeurs possibles);

- la tendance à avoir des ressources plus génériques, et donc en plus faible volume, est poussée au maximum.

L'expression "analyse non combinatoire de flux" est quasiment un pléonasme car un flux ne pourrait pas être analysé par un algorithme combinatoire, mais uniquement par un algorithme de complexité pratique linéaire, car un flux est supposé s'écouler à débit constant.

3.4.1. Du traitement de phrase au traitement de flux de caractères

Un traitement de phrase nécessite de couper le flux de caractères entre 2 phrases a priori. La suite du traitement a dès le départ une vue globale sur la phrase courante.

Définissons un traitement en flux comme un traitement des données à débit constant. Le grain du flux est l'unité minimale traitée, ici le caractère. On peut quantifier le débit en nombre d'unités traitées par unité de temps, ici en nombre de caractères par seconde, ou en Ko/s. Ajoutons au traitement de flux la caractéristique que le grain et les éléments de niveau supérieur sont traités en une passe (ou couche) unique, et non pas en couches successives (sur les tokens d'une phrase, ou sur des éléments de niveau supérieur) comme dans l'analyseur SYLEX de Patrick Constant (environ 60 couches sur les mots de la phrase).

On peut poser le problème (en le simplifiant) comme **le choix d'une chronologie de parcours** d'un espace à 2 dimensions : la dimension des tokens (l'axe syntagmatique = l'axe du flux), et la dimension des couches (et/ou des niveaux) de traitement sur le flux :

- soit (1) on traite tous les tokens d'une phrase pour chaque couche de traitement (une séquence de tokens délimitée a priori est traitée comme un tout, la phrase habituellement), c'est le traitement par couches successives sur les tokens d'une phrase :

```
pour chaque couche de traitement
      pour chaque token de la phrase
            traiter le token
```

- soit (2) on traite toutes les couches de traitement pour chaque token du flux (pas de séquence de tokens délimitée a priori à traiter), c'est le traitement par tokens successifs :

```
pour chaque token du flux
      pour chaque couche de traitement
            traiter le token
```

(NB : l'algorithme ci-dessus est seulement explicatif pour illustrer le choix entre les 2 solutions; l'articulation entre niveaux est décrite dans la section suivante)

Le traitement par couches successives (1) implique de ne pas avoir terminé certains traitements d'un token au moment de certains traitements de tokens suivants.

Le traitement par tokens successifs (2), c'est-à-dire en flux, implique d'avoir terminé tous les traitements d'un token au moment des traitements du token suivant.

Quelle est la motivation d'un traitement en flux ? C'est une motivation due aux nécessités de la **chronologie** des déductions :

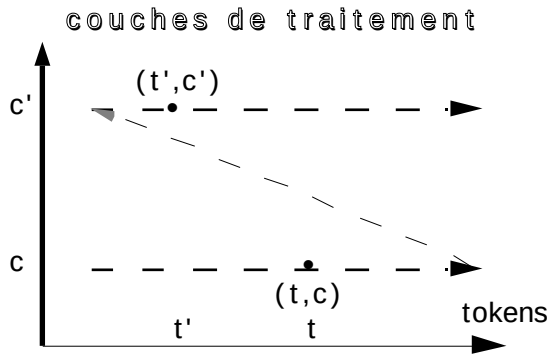
- si A=>B et B=>C, la première déduction doit être exécutée avant la deuxième;

- dans un traitement en couches successives (1), on fait l'hypothèse qu'il est possible de traiter le token t à la couche c sans avoir terminé les traitements du token t' situé avant t (ceux de la couche c'>c), autrement dit qu'on dispose de toutes les informations nécessaires (valeurs d'attributs d'éléments) pour exécuter les déductions de l'événement (t',c') sans avoir exécuté les déductions de l'événement (t,c);

- dans un traitement en flux, par tokens successifs (2), on fait l'hypothèse que, pour traiter le token t à la couche c, il faut avoir terminé tous les traitements du token t' situé avant t (dont ceux de la couche c'>c) :

analyseur 98

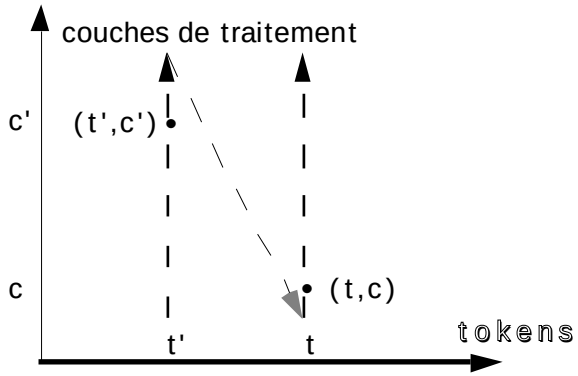
(1) traitement par couches successives
sur les tokens d'une phrase



(t,c) est chronologiquement
antérieur à (t',c')

analyseur 99

(2) traitement par tokens successifs
(= en flux, sur l'axe syntagmatique)

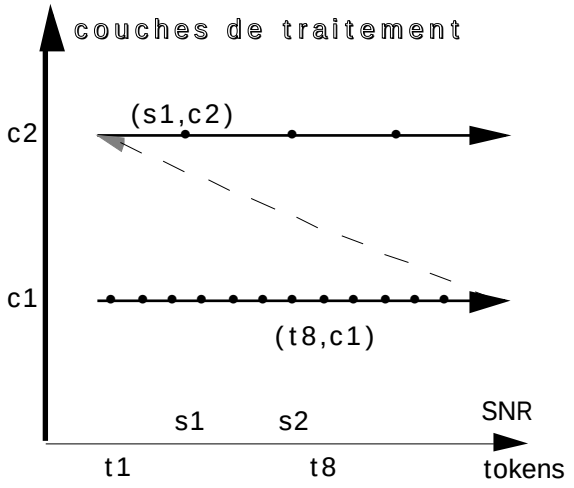


(t,c) est chronologiquement
postérieur à (t',c')

Dans l'exposé ci-dessus, les termes de "tokens" et de "couche" sont simplificateurs pour faciliter l'explication. Le terme de "tokens" est à généraliser en tous segments placés en un même point sur l'axe syntagmatique : caractère, token, syntagme, proposition, phrase, ... Le terme de "couche" est aussi à généraliser en tout traitement à un niveau de segmentation : caractère, token, syntagme, proposition, phrase, ... À titre d'illustration, voici une description analogue, mais plus proche de sa réalité, où la couche c1 est la couche de traitement des tokens, et c2 la couche de traitement des SNR; dans le traitement en flux, les tokens t1, t2 et t3 sont traités en c1, et concaténés pour constituer le SNR s1, dont le traitement en c2 commence immédiatement :

analyseur 98

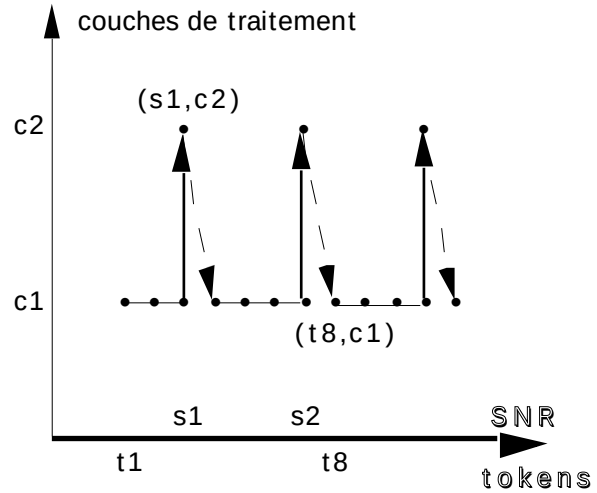
(1) traitement par couches successives
sur les tokens et les SNR d'une phrase



chronologie :
(t8,c1) puis (s1,c2)

analyseur 99

(2) traitement par tokens et SNR successifs
(= en flux, sur l'axe syntagmatique)



chronologie :
(s1,c2) puis (t8,c1)

Le traitement en flux permet qu'une déduction de couche c2 serve de prémisses à une déduction de couche c1.

Une deuxième motivation d'un traitement en flux est qu'il ne nécessite pas de définir et délimiter a priori une séquence de tokens à traiter comme un tout : la limite de phrase par exemple pourra être un résultat du calcul et non une donnée du calcul.

Remarquons que le traitement d'analyse en flux est aussi celui d'un humain en situation de réception. Notons deux points communs minimaux entre l'analyseur et l'humain en réception : la faisabilité de la chronologie des déductions et la complexité linéaire en temps des processus.

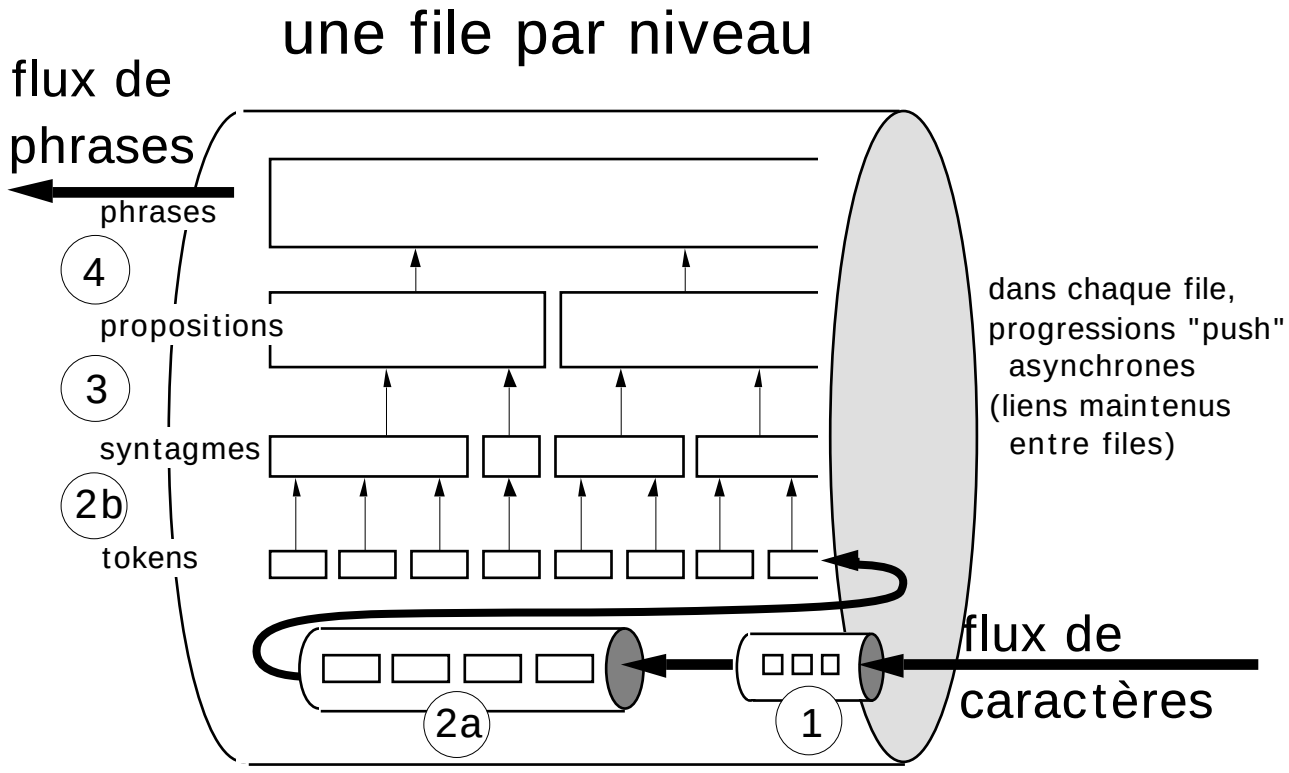
L'analyseur 98 utilise le traitement en couches successives (1) : il tokenise le flux de caractères jusqu'à une fin de phrase, puis envoie l'ensemble des tokens de cette phrase à la suite du traitement en 2 couches successives pour la phrase : 1) délimitation des SNR avec catégorisations des tokens, puis 2) mise en relation des SNR (décrit ci-dessus en 3.3).

L'analyseur 99 utilise le traitement en flux, par tokens et SNR successifs (2) : il tokenise le flux de caractères, et déclenche le traitement du token, et le traitement du SNR dès que possible, ce qui permet de traiter quasi-simultanément en flux tous les niveaux linguistiques (caractères, tokens, SNR, propositions, phrases, ...).

3.4.2. Architecture générale et articulation entre moteurs

L'analyseur est exclusivement constitué de moteurs dont chacun traite un flux d'éléments avec des règles déclaratives :

Figure 8 : Schéma global de l'architecture de l'analyseur 99



• chaque moteur est spécialisé dans le traitement d'un niveau linguistique :

fonction du moteur	niveau traité = éléments entrants	éléments produits	attributs modifiés
(1) tokenisation	caractères	tokens	(aucun)
(2a) dissoc., assoc., préétiqu. de tokens	tokens	tokens	tokens
(2b) catég. des tok. et délimit. des SNR	tokens	SNR	tokens et SNR
(3) mise en relation des SNR	SNR	propositions	SNR et propositions
(4) mise en relation des propositions	propositions	phrases	propos. et phrases

• articulation entre moteurs :

chaque moteur contrôle le moteur suivant, en montant dans la hiérarchie des niveaux :

principe général d'articulation : une action d'une règle du moteur (m) peut déclencher le moteur (m+1)

- le moteur de tokenisation (1) est placé dans la boucle de lecture du flux de caractères; les conditions portent sur 2 ou 3 caractères contigus; les actions de ses règles sont : clore un token, ouvrir un nouveau token;

la clôture du token courant provoque :

. l'envoi du token courant au début de la file des tokens

. la progression du flux de tokens dans la file des tokens

. le déclenchement des moteurs (2a) et (2b) sur ce nouveau token

- une des actions des règles des moteurs (2a) et (2b) est : ouvrir un nouveau SNR (la clôture du SNR précédent est implicite);

l'ouverture d'un nouveau SNR provoque :

. l'envoi du nouveau SNR au début de la file des SNR

. la progression du flux de SNR dans la file des SNR

. le déclenchement du moteur (3) sur ce nouveau SNR

- une des actions des règles du moteur (3) est : ouvrir une nouvelle proposition (la clôture de la proposition précédente est implicite)

l'ouverture d'une nouvelle proposition provoque :

. l'envoi de la nouvelle proposition au début de la file des propositions

. la progression du flux de propositions dans la file des propositions

. le déclenchement du moteur (4) sur cette nouvelle proposition.

D'où l'algorithme général :

```
analyseur 99 :
   répéter        (par caractère)
   |  acquérir un caractère
   |  (1) passer les règles de tokenisation sur le caractère courant
   |  concaténer le caractère courant au token courant
   jusqu'à  la fin du texte
```

Le moteur (1) applique ses règles sur des caractères :

```
(1) passer les règles de tokenisation sur le caractère courant :
      pour chaque règle
        si  la condition de la règle s'applique au caractère et ses voisins
          alors  appliquer l'action de la règle au caractère et ses voisins
```

Une des actions du moteur (1) est de clore le token courant et de l'envoyer au début de la file des tokens (voir la suite de l'algorithme ci-dessus).

3.4.3. Structures de données du flux et progression du flux dans les structures :

- le flux entre dans des files FIFO (first in - first out) analogues à des tuyaux :
  . une file de caractères,
  . une file de tokens,
  . une file de SNR,
  . une file de propositions (en cours d'implémentation),
  . une file de phrases (en cours d'implémentation);
- concrètement les files sont implémentées dans des buffers circulaires, ce qui évite les décalages dans les files, et simplifie la gestion des liens entre éléments de files différentes;
- le moteur de tokenisation (1) opère sur une file de 3 caractères (c'est suffisant pour traiter tous les cas);
- un caractère est concaténé par défaut pour produire un token qui alimente la file des tokens;
- deux moteurs agissent successivement sur la file des tokens :
  . le moteur de dissociation ou association de tokens (2a),
  . le moteur de catégorisation des tokens et de délimitation des SNR (2b);
- un token est concaténé par défaut au SNR courant, ce qui alimente la file des SNR;
- un SNR est relié avec les tokens qu'il contient (liens réciproques), et l'ensemble progresse de manière asynchrone chacun dans leur file, tout en restant reliés; ces progressions conjointes sont déclenchées dans la file des tokens par l'arrivée d'un nouveau token, et dans la file des SNR par l'arrivée d'un nouveau SNR;
- sortie du flux : l'arrivée d'un nouveau SNR pousse un SNR en sortie, avec tous ses tokens.

Remarque : l'entrée est un flux de caractères, mais la sortie est un flux d'éléments du plus haut niveau traité par l'analyseur, pour l'instant un flux de SNR, plus tard un flux de propositions, puis de phrases, puis de paragraphes, ...

3.4.4. Flux, règles déclaratives, moteur

Un moteur produit des transformations sur un flux d'éléments, et un paquet de règles est appliqué à un élément courant (le front de règle) et ses voisins précédents. Un élément est un caractère, un token, un SNR, une proposition, ...

Toutes les règles sont de la forme :

conditions sur des attributs de l'élément courant et sur des attributs de ses voisins précédents
=> **actions** sur des attributs de l'élément courant et sur des attributs de ses voisins précédents

Le terme "voisin" est à entendre sur les 2 axes : voisin sur l'axe syntagmatique et voisin sur l'axe des niveaux.

On peut établir une analogie avec un système expert :

système expert	<--->	ensemble moteur + règles + flux d'éléments
moteur	<--->	moteur

règles de déduction	<--->	règles de modification des attributs
base de faits	<--->	flux d'éléments entrants

On peut donc comparer l'analyseur à un ensemble coordonné de systèmes experts.

3.4.5. Structure et algorithme du moteur générique

Entendons par moteur générique, le concept de moteur à partir duquel tous les moteurs réels sont dérivés.

Le moteur générique est un "**si** condition **alors** action" généralisé (et répétitif sur les règles), dans lequel "condition" et "action" sont dissociés et généralisés dans le moteur, mais réassociés et particularisés dans chaque règle déclarative.

Dans le moteur, la fonction booléenne "condition" consiste en l'inventaire des définitions de toutes les conditions, et, de même, la procédure "action" consiste en l'inventaire des définitions de toutes les actions; j'entends par "définition", l'association d'un code du formalisme avec une condition ou une action.

Appliquer un paquet de règles à quelques éléments (le plus souvent contigus, mais pas nécessairement) consiste à appliquer successivement chaque règle du paquet à ces éléments. Comme plusieurs règles sont applicables, et comme l'application d'une règle peut rendre une règle suivante applicable, tout le paquet doit passer sur les éléments.

Cet algorithme est en temps constant pour un élément, dépendant seulement du nombre constant de règles du paquet, et de l'empan constant de chaque règle.

Algorithme du moteur générique pour un élément du flux (en un point de l'axe syntagmatique) :

```
pour chaque règle du moteur
   si  la condition de la règle s'applique à l'élément et ses voisins
      alors  appliquer l'action de la règle à l'élément et ses voisins
```

Appliquer une règle en un point de l'axe syntagmatique (le front de règle) consiste à évaluer la fonction booléenne sur les éléments de l'empan de la règle (opérateur **et**), et si la fonction est vraie, à appliquer les actions sur chaque élément de l'empan.

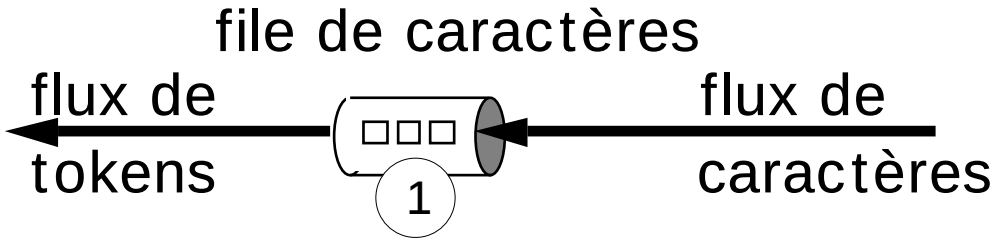
La fonction booléenne sur un élément consiste en l'inventaire des définitions de toutes les conditions sur un élément, en associant chaque fois un code du formalisme avec une condition sur les attributs de l'élément. Il est aisé d'implémenter les opérateurs booléens **et** et **non** par des appels récursifs de cette fonction. Nota bene : l'opérateur booléen **ou** est réalisé implicitement par plusieurs règles successives.

La procédure implémentant l'action sur un élément consiste en l'inventaire des définitions de toutes les actions sur les attributs de l'élément, en associant chaque fois un code du formalisme avec une action.

Par "attribut de l'élément", il faut entendre de manière extensive : "attribut de l'élément ou de tout autre élément du même niveau ou d'un autre niveau qui lui est relié" (relié est à prendre en un sens très général).

3.4.6. Propriétés particulières des moteurs 1 à 4

1) moteur de tokenisation

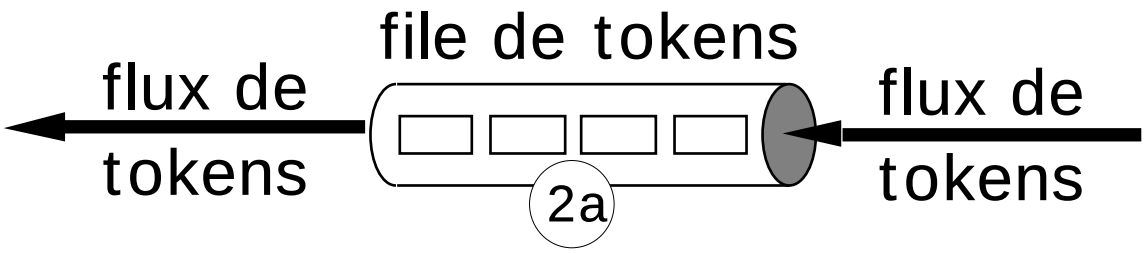


fonction : découper le flux de caractères en tokens
structures de données concernées : file des caractères et token en cours de construction
éléments entrants : caractères
éléments sortants : tokens
empan des règles : 3 caractères
nombre de règles : 17 pour le français, 14 pour l'anglais
opérateurs des conditions sur un caractère : majuscule, chiffre, appartenance à un ensemble de caractères, avec les opérateurs booléens **non et**
opérateurs des actions : clore, ouvrir le token courant sur le caractère courant, insérer une balise de fin de phrase, ou une balise de fin de paragraphe.

règles :	espace + non espace		*conditions sur les caractères*
=>	+ [		*action : ouvrir un token*
	non espace + espace		*conditions sur les caractères*
=>	] +		*action : clore un token*

clore le token courant ==>
- pousser d'un cran les tokens dans leur file (liens maintenus entre files)
- envoyer le token courant au début de la file des tokens
- déclencher les moteurs (2a) et (2b) sur ce nouveau token

2a) moteur de dissociation, association, préétiquetage par défaut des tokens



fonctions : dissocier les amalgames en 2 ou 3 tokens
séparer les pronoms atones postposés (d'où 2 tokens)
associer 2 à 4 tokens d'après le lexique ("peu à peu" --> "|peu_à_peu")
associer une abréviation avec son point, et effacer la balise de fin de phrase
associer et étiqueter 2 nombres en chiffres
associer et étiqueter 2 nombres en lettres
associer et étiqueter 2 noms propres (par le lexique ou la majuscule)

étiqueter les nombres en chiffres
étiqueter le token d'après le lexique des mots grammaticaux
étiqueter les nombres en lettres
étiqueter le token d'après la base de formes verbales
étiqueter le token d'après la base de finales

structure de données concernée : file des tokens

éléments entrants : tokens

éléments sortants : tokens dissociés ou associés

empan des règles : 1 à 4 tokens

nombre de règles : 68 pour le français, 52 pour l'anglais

opérateurs des conditions sur un token : terminaison par une finale donnée, nombre en chiffres, nombre en lettres, nom propre du lexique, initiale majuscule, graphie appartenant à un ensemble de graphies donné, avec les opérateurs booléens **non et**

opérateurs des actions sur un token : voir les fonctions ci-dessus

éléments dont les attributs sont modifiés : tokens

Exemples de règles de dissociation :

1 token --> 2 tokens du -> \de>p + le>d/ms doesn't -> does + n't
darlos -> dar + los donne-le -> donne + -le

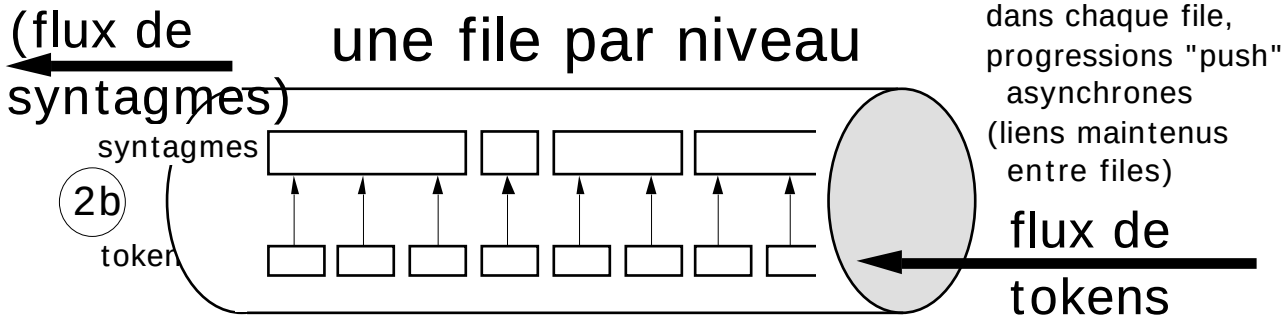
Exemples de règles d'association :

2 à 4 tokens --> 1 token peu + à + peu -> |peu_à_peu a + priori -> |a_priori
10 + 000 -> |10_000 vingt + et + un -> |vingt_et_un
M. + Albert + Dupont -> |M._Albert_Dupont

Exemples de préétiquetage :

- par le lexique des mots grammaticaux dans > préposition
- par la base de formes verbales aimerait > V/s3|aimer.1
- par la base de finales -ation > S/fs -iblement > j

2b) moteur de catégorisation des tokens et de délimitation des SNR



fonctions : délimiter et typer les SNR d'après les tokens
compléter l'étiquetage des tokens d'après les types de SNR

structures de données concernées : file des tokens, file des SNR

éléments entrants : tokens

éléments sortants : SNR

empan des règles : 1 à 3 tokens

nombre de règles : 54 pour le français, 36 pour l'anglais

opérateurs des conditions sur un token : graphie appartenant à un ensemble de graphies donné, catégorie appartenant à un ensemble de catégories donné (avec contrainte éventuelle sur un attribut), type du SNR courant, accord possible avec le début du SNR courant, avec les opérateurs booléens **non et**

opérateurs des actions sur un token : ouvrir un nouveau SNR, typer un SNR, catégoriser un token, propager l'accord dans le SNR courant,

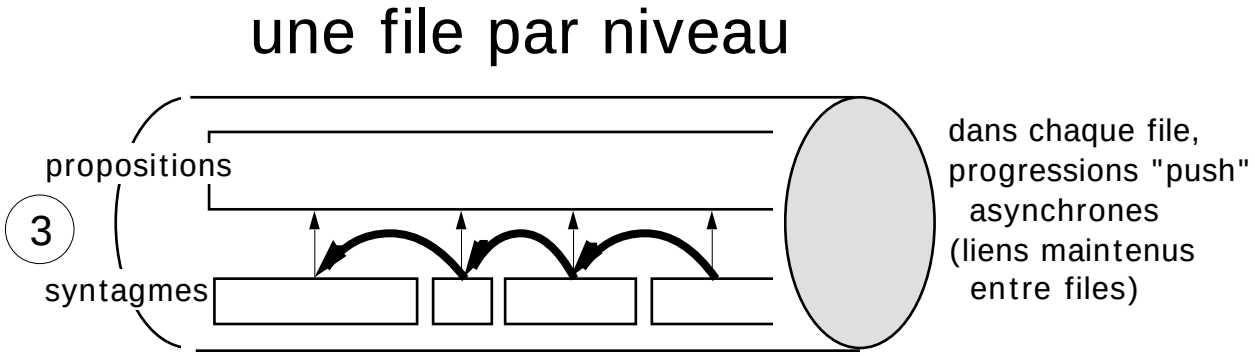
éléments dont les attributs sont modifiés : tokens et SNR

Exemples de règles :

ouvrir un SNR (non typé) préposition => ["il" => [typer_SNR(V)
typer un SNR par un token "dans" => typer_SNR(N) "ne" => typer_SNR(V)
typer un token par un SNR "le" & type(V) => pronom objet
"proche" & type(V) => adjectif attribut

ouvrir un SNR ==>
- pousser d'un cran les SNR dans leur file (liens maintenus entre files)
- envoyer le nouveau SNR au début de la file des SNR
- clore le SNR précédent et l'envoyer au moteur (3)

3) moteur de mise en relation des SNR dans les propositions



fonctions : relier les SNR dans les propositions
(dépendances et coordinations des actants et circonstants)
délimiter et typer les propositions
compléter l'étiquetage des SNR d'après les relations entre SNR
(compléter l'étiquetage des tokens d'après les types de SNR)

structures de données concernées : file des SNR, file des propositions

éléments entrants : SNR

éléments sortants : propositions

empan des règles : 1 à 3 SNR

opérateurs des conditions sur un SNR : type du SNR courant, contiguïté avec un SNR de tel type, présence d'un SNR de tel type dans telle mémoire, accord possible du SNR courant avec un SNR dans telle mémoire, relation de dépendance du SNR courant avec un SNR de tel type, isomorphisme de coordination du SNR courant avec un SNR de telle mémoire, avec les opérateurs booléens **non et**

opérateurs des actions sur un SNR : placer un SNR dans telle mémoire, relier un SNR (dépendance ou coordination) avec un SNR placé dans telle mémoire, désignation du SNR à mettre en mémoire ou du SNR à relier (différent du SNR courant), typer une proposition, typer un SNR, oublier un SNR dans une mémoire, catégoriser un token

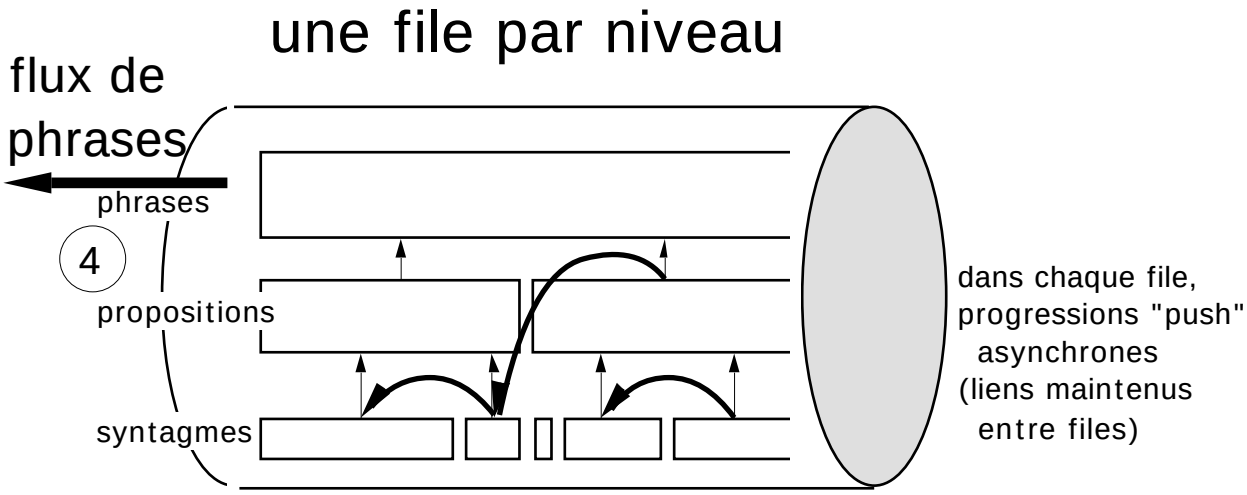
éléments dont les attributs sont modifiés : SNR et propositions

Exemples de règles :

mettre un SNR en attente de relation (dépendances et coordinations)

relier un SNR (dépendance ou coordination) avec un SNR placé en mémoire
compléter l'étiquetage des SNR d'après les relations entre SNR
ouvrir et clore une proposition
ouvrir une proposition ==>
- pousser d'un cran les propositions dans leur file (liens maintenus entre files)
- envoyer la nouvelle proposition au début de la file des propositions
- déclencher le moteur (4) sur cette nouvelle proposition

4) moteur de mise en relation des SNR et propositions dans une phrase



fonctions : relier les propositions à leur régissant (proposition ou SNR)
structures de données concernées : file des SNR, file des propositions
éléments entrants : SNR et propositions
éléments sortants : SNR et propositions
(moteur en cours de développement)
éléments dont les attributs sont modifiés : SNR et propositions

Exemples de règles :
- mettre une proposition en attente de subordonnée, de coordonnée
- relier une proposition avec une proposition ou un SNR placé en mémoire
ouvrir une phrase ==>
- pousser d'un cran les phrases dans leur file (liens maintenus entre files)
- envoyer cette nouvelle phrase au début de la file des phrases

3.4.7. Fonctions génériques

À tous niveaux, les fonctions sont les mêmes :
- regrouper des éléments en un groupe d'éléments (l'élément du niveau supérieur)
- étiqueter (ou typer) des éléments
- relier des éléments à l'intérieur de l'élément du niveau supérieur.

Un regroupement d'éléments peut se faire suivant 2 processus :
- repérer des frontières en détectant des différences entre éléments contigus :
. regrouper des caractères en un token : différence entre 2 caractères contigus
. regrouper des tokens en un SNR : différence entre 2 tokens contigus

- relier des éléments dans l'élément du niveau supérieur, ce qui le définit comme étant constitué des éléments reliés à un même élément central :

. relier des SNR dans une proposition, définie comme étant constituée des SNR reliés à son verbe.

Il serait envisageable de relier des tokens dans un SNR, défini comme étant constitué des tokens reliés à son élément central, nom ou verbe.

L'étiquetage (ou typage, ou catégorisation) d'un élément est effectué sous la contrainte du type d'un élément inclus de niveau inférieur, ou sous la contrainte du type de l'élément englobant de niveau supérieur.

Le processus de mise en relation des SNR par mémoires (cf. 2.3.2) est générique et transposable à d'autres niveaux.

3.4.8. Une voie d'optimisation

Les algorithmes présentés ci-dessus sont de complexité linéaire et permettent un traitement du flux à débit constant, mais la complexité dépend aussi linéairement de l'empan des règles et du nombre de règles : si on double le nombre de règles, le débit est divisé par 2. Dans l'état actuel, toutes les règles d'un moteur passent sur un élément, car plusieurs règles sont applicables à cet élément. Une voie d'optimisation sera de ne pas passer toutes les règles mais seulement celles qui sont applicables. Comment savoir à l'avance quelle règle est susceptible d'être applicable ? La question est posée mais non résolue. La complexité restera linéaire, mais le débit pourra être augmenté.

4. Validation et valorisation des recherches

4.1. Validation opératoire (et donc théorique) : première place à l'action d'évaluation GRACE

L'action d'évaluation GRACE[14] est une composante du programme thématique "Cognition, Communication intelligente et Ingénierie des langues", à l'initiative de J. Mariani (Limsi) et de R. Martin (INaLF). Cette action vise à "la mise en place d'un paradigme d'évaluation pour les analyseurs morpho-syntaxiques et syntaxiques et la constitution d'un premier noyau de données réutilisables pour l'évaluation de systèmes linguistiques d'analyse du français".

Cette action a commencé en octobre 1995, et la dernière adjudication s'est terminée en octobre 98. Les derniers (avril 99) résultats publiés, le 6/11/98 sur `http://www.limsi.fr/TLP/grace/`, sont ceux de l'évaluation absolue; ils sont officiels et définitifs, mais très incomplets : seuls 4 systèmes sont nommés sur les 13 qui ont participé jusqu'à la fin, ceux du LIMSI et de l'INALF, organisateurs, et ceux du GREYC et du LATL, les 2 seuls qui ont validés leurs résultats et qui ont accepté de les rendre publics.

Les participants ont regroupé les meilleurs spécialistes de l'analyse automatique du français :

- 8 laboratoires français :
  - **GREYC**, Caen, Jacques Vergne
  - CNET, Dpt LAA/EIA/AIA, Lannion, Christine Chardenon
  - Laboratoire CLIPS, IMAG, Grenoble, Damien Genthial
  - Equipe CRISTAL-GRESEC, Université Stendhal, Grenoble, Geneviève Lallich-Boidin
  - Laboratoire Informatique d'Avignon avec Laboratoire Parole et Langage d'Aix-en-Provence, Thierry Spriet
  - **INalF** - CNRS (Nancy), Josette Lecomte (organisateur-participant, donc non classé)
  - **LIMSI** (Orsay), Stéphane Ferrari (organisateur-participant, donc non classé)
- 2 laboratoires de Genève :
  - Laboratoire **LATL**, Université de Genève, Eric Wehrli
  - ISSCO / Université de Genève, Gilbert Robert
- 1 laboratoire québécois:
  - Centre d'innovation en technologies de l'information, CITI, Québec, Jean-Marc Jutras
- 3 laboratoires allemands :
  - [Institut für Angewandte Informationsforschung (IAI), Saarebrücken, Heinz-Dieter Maas]
  - [Institut für Linguistik/Romanistik (Universität Stuttgart) & Institut für maschinelle Sprachverarbeitung (IMS), Stuttgart, Achim Stein]
- 8 entreprises de tailles variées, françaises et américaines :
  - INGENIA-Langage Naturel S.A., Paris, Patrick Constant
  - [MEMODATA, éditeur du logiciel Dicologique, Caen, Dominique Dutoit]
  - [GSI-Erli, Paris, Hugues de Mazancourt]
  - [T.GID : société du groupe technologies, éditeur du logiciel SPIRIT, Paris, Patrick Mordini]

[14] GRACE : **G**rammaires et **R**essources pour les **A**nalyseurs de **C**orpus et leur **E**valuation
`http://www.limsi.fr/TLP/grace/`

[Société Synapse Développement, éditeur du logiciel Cordial, Toulouse, Dominique Laurent]
IBM France, Paris, Marie-Hélène Antoni
Centre de Recherche Rank Xerox, Grenoble, Jean-Pierre Chanod
[AT&T Bell Laboratories, New Jersey, USA, Evelyne Tzoukermann]
(NB : les participants entre [] ont abandonné avant la fin)

Le protocole global de cette action était le suivant : un corpus de 100 000 mots (journal Le Monde et littérature) a été étiqueté manuellement (à chaque mot sa catégorie grammaticale); ce corpus a été noyé dans un corpus de 500 000 mots, ce qui a donné un corpus de 600 000 mots, qui a été transmis aux participants (par ftp); ces derniers en ont fait l'étiquetage automatique ("tagging") sous 48h; les résultats des participants ont ensuite été comparés automatiquement à l'étiquetage manuel, d'où une mesure de distance entre l'étiquetage manuel et les étiquetages automatiques des participants (mesure suivant deux critères : la décision, ou capacité de choisir une seule étiquette (bonne ou mauvaise) ou de la déduire du contexte, et la précision, ou capacité de choisir comme l'étiqueteur humain, quand un choix a été fait).

Il y a eu trois phases : entraînement sur corpus d'octobre 95 à août 96, répétition générale du protocole en septembre 96, tests finaux en décembre 97; ces phases ont été accompagnées de périodes de discussions (dont j'ai été le principal animateur : 77 messages envoyés d'octobre 95 à juillet 98) par courrier électronique entre participants et comité de coordination[15], pour construire un consensus sur le jeu d'étiquettes (11 catégories principales, et 311 étiquettes différentes), sur les spécifications d'étiquetage, sur le protocole d'évaluation, et sur les modalités scientifiques, techniques et organisationnelles du projet.

Des obstacles nombreux, imprévus et d'origine théorique ont surgi tout au long du projet, par exemple : étant donné qu'il n'y a pas de définition a priori du mot, l'unité traitée (le "token") a été différente pour chaque participant, et il a fallu "réaligner" les unités de chaque participant avec les unités du corpus de référence; comme chaque participant avait son propre jeu d'étiquette, il a fallu trouver une méthode de transcodage vers le jeu du corpus étiqueté manuellement : le comité de coordination a proposé à chaque participant de construire une "table de correspondance" entre les deux jeux d'étiquettes, permettant d'affecter une étiquette GRACE pour chaque étiquette du participant, pour éviter aux participants de modifier leur système, mais cela revenait à faire l'hypothèse que cette mise en correspondance était possible, ce qui n'est vrai que partiellement. C'est pour cela que, comme certains participants, j'ai écrit une fonction de transfert qui opère le changement de tokenisation et le changement d'étiquettes (environ 1500 lignes). De plus, avec les corpus étiquetés manuellement des essais, j'ai pu reproduire le protocole d'évaluation et préparer les tests finaux en ayant continuellement les résultats d'évaluation. Pour que mes résultats soient soumis à la même "moulinette" d'évaluation que les autres participants, j'ai fourni aux organisateurs une table de correspondance (appelée "table identité") où toute étiquette était remplacée par elle-même.

L'objet même du projet a évolué : il s'agissait au début d'évaluer la capacité du système à choisir une étiquette parmi une liste proposée par les organisateurs pour chaque mot; cette opération est couramment appelée "désambiguïsation", car le "tagging" est souvent conçu comme un choix parmi une liste d'origine lexicale ("désambiguïser" = sortir de l'"ambiguïté lexicale"[16]), alors qu'on sait d'une part qu'un lexique est toujours doublement incomplet : incomplet car des mots n'y sont pas, et incomplet car des catégories possibles manquent, et d'autre part qu'un texte contient des noms propres et des néologismes qui ne peuvent être dans un dictionnaire. Cette liste proposée par les organisateurs devait

[15] Ces messages constituent ensemble un document de 95 pages (sans saut de page, 230 Ko pour 77 messages).
[16] Le terme "ambiguïté" est mal choisi, car il n'y a ambiguïté que pour l'humain, alors qu'ici, il s'agit d'un choix à faire pour la machine.

être constituée à partir du lexique MULTEXT. Mais les modifications du jeu d'étiquettes GRACE imposaient une modification du lexique, ce qui a entraîné que le lexique modifié n'a pas été prêt à temps pour les essais. Or, j'avais demandé avant les essais qu'il n'y ait aucune ressource imposée, pour que les participants aient le choix complet des moyens (ressources lexicales, déductions locales statistiques ou symboliques, mise en relation ou pas) pour réaliser une même tâche d'étiquetage en partant du texte brut. Aux essais, il avait été finalement convenu que les participants auraient le choix : soit partir du texte préétiqueté avec MULTEXT, soit partir du texte brut. Mais aux essais, au moment de la réception du corpus à étiqueter, les participants n'ont reçu que le corpus brut, et pas le corpus préétiqueté, mais nous avions tous nos propres ressources lexicales, et nous avons tous étiqueté le corpus brut. Donc aux essais, l'objectif de GRACE a changé : il ne s'agissait plus d'évaluer les capacités de "désambiguïsation" d'un système mais plus globalement ses capacités d'étiquetage : l'étiquetage n'était plus de facto vu comme la seule "désambiguïsation". Dans le test final, le protocole des essais a été maintenu : étiquetage des textes bruts.

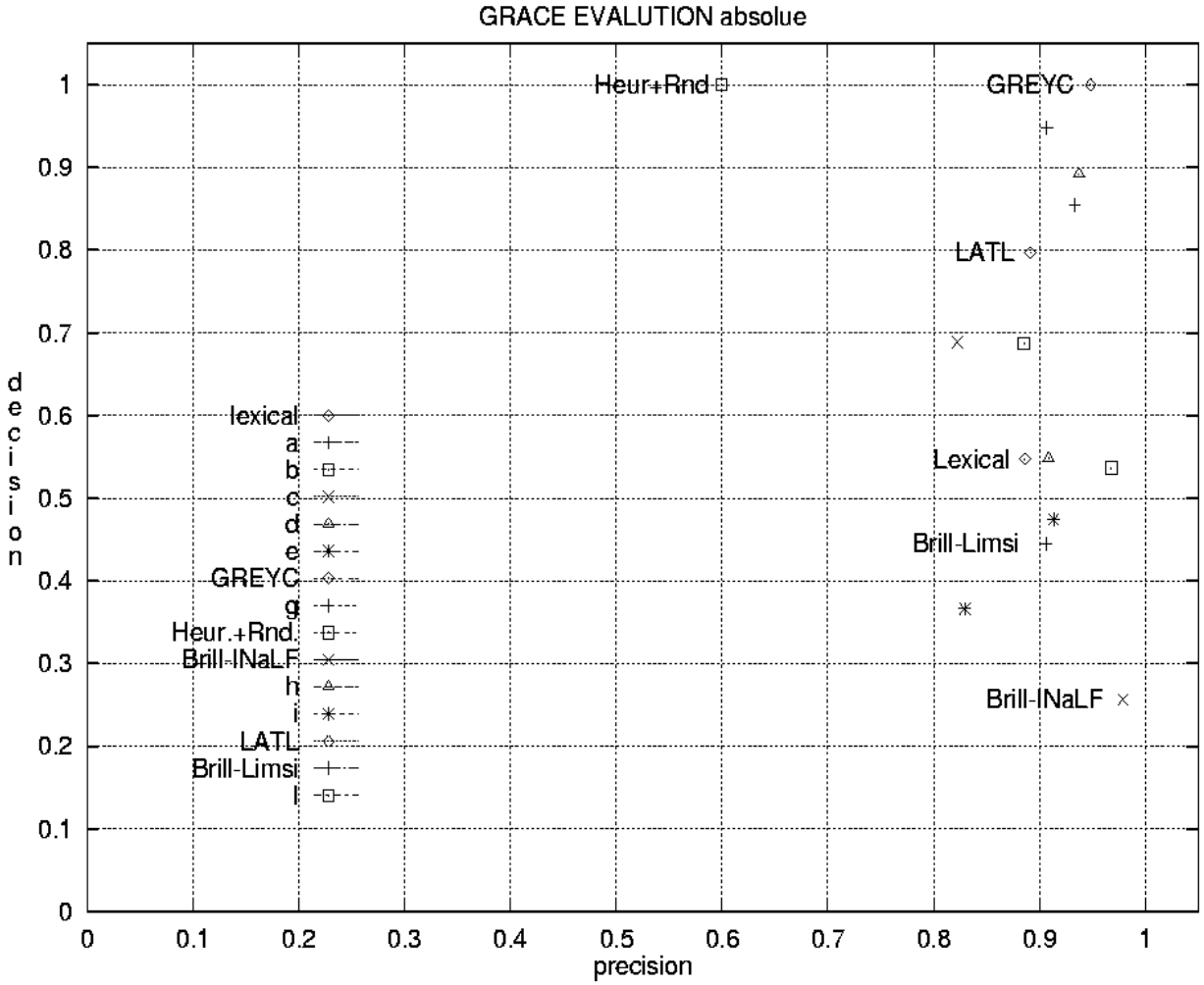
On a pu constater l'importance des bases théoriques des systèmes en lice (voir ci-dessus en 3.2, et l'article présenté à TALN'98 : Regards Théoriques sur le “Tagging”) : ceux qui utilisaient le syntagme non récursif (un niveau hiérarchique au dessus du mot) ont été avantagés par rapport à ceux qui concevaient la phrase comme une simple suite de mots, car le type du syntagme et le type de ses éléments sont liés (ces liens permettent de faire des déductions), et la déduction contextuelle n'est sûre qu'à l'intérieur d'un syntagme; d'autre part, certaines catégories et certains attributs de mots ne peuvent être affectées qu'en établissant une relation avec un mot éloigné (sujet - verbe par exemple) : il faut donc analyser la phrase pour étiqueter certains mots (2 à 3 %); les participants faisant une analyse plus complète ont donc eu de meilleurs résultats. De plus, le problème est classiquement mal posé comme un choix parmi des catégories équiprobables (ce qui est faux), alors qu'il vaut mieux poser la catégorie la plus probable comme catégorie initiale par défaut (dans le lexique), corrigée éventuellement par le contexte; j'ai fait l'expérience d'étiqueter uniquement par les catégories par défaut, sans aucune déduction locale, et on obtient 100% en décision et plus de 80% en précision (sur le jeu complet de 311 étiquettes différentes), ce qui est déjà un bon résultat avec peu de ressources.

Pour la première fois, des étiqueteurs et des analyseurs ont été comparés sur une même tâche dans des conditions expérimentales identiques, ce qui a permis de sortir des revendications de performances purement individuelles ne permettant aucune comparaison à cause des conditions expérimentales différentes.

Évaluation absolue et évaluation relative :

L'évaluation "absolue" (publiée le 6/11/98 - voir graphique ci-dessous) est faite sur le jeu d'étiquettes GRACE : les sorties du participant sont transférées de son jeu vers le jeu GRACE par sa "table de correspondance", puis comparées au corpus d'évaluation étiqueté manuellement par Josette Lecomte dans le jeu GRACE.

L'évaluation "relative" (non encore publiée 6 mois après la publication de l'évaluation absolue) est faite sur le jeu d'étiquettes du participant : les sorties du participant sont comparées au corpus d'évaluation étiqueté manuellement par Josette Lecomte dans le jeu GRACE, après son transfert du jeu GRACE vers le jeu du participant en appliquant sa "table de correspondance" à l'envers. L'évaluation "relative" pour le GREYC est identique à l'évaluation "absolue", car, ayant assumé moi-même le transfert, mon jeu d'étiquettes (celui dans lequel les résultats ont été fournis) est identique au jeu GRACE.



Commentaires sur l'évaluation absolue et sur les définitions de la "décision" et de la "précision" :

- selon la définition des organisateurs, la "décision" est le pourcentage de tokens qui ont **une étiquette unique dans le jeu GRACE**, donc **après** le transfert par la "table de correspondance"; donc, si **une** étiquette du participant correspond à **plusieurs** étiquettes GRACE, tous les tokens ayant **cette étiquette unique** après traitement du participant obtiennent **plusieurs** étiquettes GRACE après transfert et sont donc systématiquement comptés comme tokens "non décidés"; la "décision" dans l'évaluation absolue est donc d'abord une mesure de la finesse du grain du jeu du participant comparativement au jeu GRACE : si pour certaines catégories, le jeu du participant ne comporte pas certains attributs codés dans le jeu GRACE (cas fréquent, car le jeu GRACE est très fin), alors tous les tokens concernés par ces catégories sont systématiquement exclus des tokens décidés : par exemple, si un participant ne code pas le mode des verbes, tous les verbes ont autant d'étiquettes GRACE que de modes possibles (c'est le paradigme du choix poussé à l'absurde : le non-calculé est compté comme non-choisi); c'est donc seulement à finesses comparables que la "décision" mesure ce qu'elle est censée mesurer : le comportement du tagger du participant dans sa capacité à sortir une seule étiquette par token; tout ceci explique que 6 participants sur 13 ont une faible "décision", allant de 25% à 55%; en conclusion, cette définition de la "décision" est incompatible avec l'ambition de comparer des taggers ayant des jeux différents au moyen de "tables de correspondance" entre 2 jeux;

- selon la définition des organisateurs, la "précision" est, parmi les "tokens décidés", le pourcentage de tokens qui ont leur unique étiquette identique à celle de l'étiquetage manuel (dans le jeu GRACE); la "précision" est donc une sous-catégorisation de la "décision"; pour les participants ayant une "décision"

faible, cette mesure de "précision" ne porte que sur un sous-ensemble des tokens, ceux ayant certaines catégories de tokens dont la table de correspondance donne une étiquette GRACE unique; ce sous-ensemble inclut les ponctuations, conjonctions, prépositions, interjections, tokens dont la catégorie est déterminée lexicalement, sans déduction locale, et peut exclure tous les verbes (si le jeu du participant ne comporte pas le temps ou le mode par exemple), ou tous les noms (si le jeu du participant ne comporte pas le genre ou le nombre par exemple), etc.; ceci explique les bonnes "précisions" obtenues, surtout à "décision" faible. Notons que le produit "décision" par "précision" donne la proportion de tokens étiquetés comme dans le corpus étiqueté manuellement.

Commentaire plus général sur cette métrique d'évaluation : la "décision" se situe dans le paradigme de la "désambiguïsation", comme mesure de la capacité de l'ordinateur à choisir, et donc en posant le problème de l'étiquetage comme un problème d'automatisation du choix de l'étiquette parmi des étiquettes possibles exhaustivement énumérées, alors que nous le posons comme un problème de calcul à partir de valeurs initiales par défaut. C'est ce qui nous a permis de sortir systématiquement une étiquette unique, et d'obtenir une "décision" de 100%.

Quelques propositions pour une autre métrique d'évaluation :

- considérer un tagger comme un outil à calculer et pas un outil à choisir, ce qui invalide le concept de "décision";
- calculer la précision comme une fonction de distance entre l'étiquette du participant (même multiple) et l'étiquette manuelle (il s'agit de définir une métrique dans un espace à plusieurs dimensions);
- compter un attribut non calculé non en multipliant les étiquettes suivant la combinatoire de ses valeurs mais en diminuant la précision dans la fonction d'évaluation de distance;
- il faut à terme diminuer, déplacer ou annuler l'intervention humaine : la diminuer et la déplacer par exemple en calculant automatiquement des distances entre participants, puis en comparant manuellement des participants voisins sur des échantillons contenant des tokens taggés différemment et pointés automatiquement; annuler l'intervention humaine sera possible en mettant au point des méthodes de comparaison automatique des participants entre eux et non plus par rapport à un étiquetage manuel.

Les résultats du GREYC sont de 100% en "décision" et de 94,5% environ en "précision" (produit "décision" * "précision" = 94,5%); les quatre suivants sont à 95% * 90,5% = 86% (Ingénia?), 89% * 93,5% = 83,2% (CNET? ou X?), 85,5% * 93% = 79,5% (X? ou CNET?) et 89% * 80% = 71,2% (LATL). Les suivants sont à moins de 70% en "décision" (produit "décision" * "précision" de 61% à 25%).

Cette première place constitue une très importante valorisation de l'analyseur et de ses bases théoriques linguistiques et algorithmiques, et montre le niveau de ses performances par rapport aux meilleurs spécialistes de l'analyse automatique du français. De plus, comme la transposition à d'autres langues exigera plus un travail d'ingénieur que de chercheur, le GREYC détient avec mes recherches le meilleur paradigme existant pour l'analyse syntaxique automatique de langue. Cette première place, ainsi que ma participation intensive aux discussions du projet me situe, ainsi que le GREYC, aux tous premiers rangs de la communauté française et francophone du TAL syntaxique.

Exemple d'entrée et sortie de l'analyseur du GREYC

• en entrée :

```
Une petite centaine de travailleurs de Renault-Vilvorde s'étaient rassemblés, dimanche 29 juin, en fin d'après-midi, devant les grilles de l'usine en grève.
```

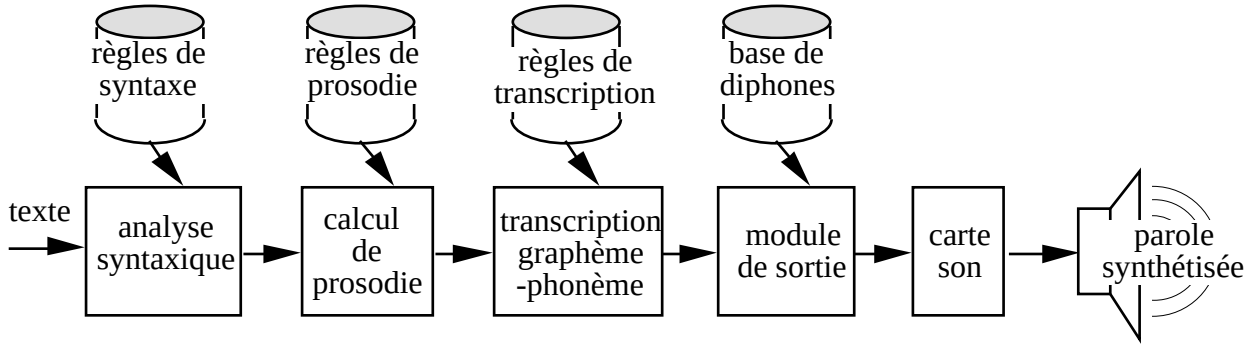
• en sortie, après segmentation et étiquetage :

La tokenisation très fine est celle convenue pour GRACE. Les syntagmes nominaux ou verbaux sont mis en évidence, comme segment dans lequel la déduction contextuelle est sûre.

```
                étiqu. étiqu.
        token > GREYC  GRACE

          Une > d/fs3. Da-fs-i
        petite > a/fs3. Afpfs
     centaine > S/fs3. Ncfs

           de > p/.... Sp
 travailleurs > S/mp3. Ncmp

           de > p/.... Sp
      Renault > S/.s3π Npms/1.3
            - > S/.s3π Npms/2.3
     Vilvorde > S/.s3π Npms/3.3

            s > m/mp3. Px3mp--/1.2
            ' > m/mp3. Px3mp--/2.2
     étaient > X/.p3. Vaii3p-
  rassemblés > z/mp3. Vmps-pm

            , > [/.... F

     dimanche > S/ms3. Ncms
           29 > E/ms3O Ak-ms
         juin > S/ms3. Ncms

            , > [/.... F

           en > p/.... Sp
          fin > S/.s3. Ncms

            d > p/.... Sp/1.2
            ' > p/.... Sp/2.2
       après > S/ms3. Ncms/1.3
            - > S/ms3. Ncms/2.3
        midi > S/ms3. Ncms/3.3

            , > [/.... F

       devant > p/.... Sp
          les > d/fp3. Da-fp-d
      grilles > S/fp3. Ncfp

           de > p/.... Sp
            l > d/fs3. Da-fs-d/1.2
            ' > d/fs3. Da-fs-d/2.2
        usine > S/fs3. Ncfs

           en > p/.... Sp
        grève > S/fs3. Ncfs

            . > ./.... F
```

4.2. Valorisation industrielle et validation de concepts sur la prosodie : direction scientifique du Projet Synthèse Vocale Kali

La synthèse vocale consiste à produire automatiquement une voix synthétique à partir d'un texte (menu de logiciel, journal, roman, etc.). La qualité première de cette voix doit être son intelligibilité (y compris à débit double ou triple), ensuite elle doit être agréable (mais l'imitation parfaite de la voix humaine n'est pas prioritaire). Pour atteindre cet objectif, il faut calculer automatiquement et en temps réel (pour synthétiser à un débit constant, et égal ou supérieur au débit humain) la durée, la hauteur et l'intensité de chaque syllabe (paramètres acoustiques de la "prosodie"), l'emplacement et la durée des pauses, ce qui nécessite une analyse syntaxique rapide, complète (catégorie des mots pour résoudre les homographes hétérophones, et relations entre groupes accentuels pour délimiter les groupes prosodiques, et placer et quantifier les pauses) et de complexité linéaire (durée stable de l'analyse). Il faut ensuite transcrire les lettres en phonèmes, et produire les diphones (34*34 suites de 2 demi-phonèmes, extraits de signal de parole) concaténés, en leur affectant les paramètres acoustiques calculés, et enfin envoyer le signal synthétique à la carte-son. Voici la chaîne des traitements :



Ce projet est exemplaire pour les raisons suivantes : il va de la recherche fondamentale à la commercialisation, et les conditions sont réunies pour produire une parole synthétique très intelligible (la plus intelligible du marché), grâce au calcul de la prosodie à partir de l'analyse syntaxique.

Les 2 partenaires publics sont : le GREYC (analyse syntaxique et calcul de la macro-prosodie) et l'ELSAP (calcul de la micro-prosodie, conversion graphème-phonème, base de diphones).

Les 2 partenaires privés sont : ELECTREL, PME caennaise (expérience de la synthèse vocale, commercialisation), et le Club MicroSon, association d'utilisateurs aveugles (cahier des charges, tests auprès des utilisateurs, commercialisation).

La convention a été signée sur 3 ans, de 96 à 98 (durée de la thèse de Gérald Vannier), avec cofinancement FEDER, et cofinancement par les 2 partenaires privés. Les délais prévus ont été tenus : la commercialisation de la synthèse du français a commencé au printemps 1999 (le lancement officiel du produit, nommé KALI, a eu lieu le 9 avril 99 à l'université de Caen, en présence de la Présidente de l'université et de la presse).

La synthèse concurrente est celle du CNET de Lannion (Proverbe diffusée par Élan Informatique), mais ses caractéristiques privilégient la compréhension immédiate à débit normal et l'utilisation de courte durée par téléphone par des utilisateurs non entraînés, alors que notre synthèse privilégie l'intelligibilité à débit rapide et l'utilisation de longue durée par des utilisateurs entraînés, le plus souvent des aveugles travaillant sur ordinateur.

Ma participation à ce projet est à plusieurs niveaux :

- l'intégration de l'analyseur dans un produit industriel,

- la confirmation d'hypothèses complètement originales sur l'isomorphisme entre l'arbre des dépendances et la prosodie : la prosodie permet à l'auditeur de reconstruire l'arbre (nous avons obtenu une des meilleures prosodies du marché, qui permet une lecture intelligible et agréable de journaux et de romans),

- l'encadrement de Gérald Vannier (avec Anne Lacheret de l'ELSAP), le doctorant dont les recherches portent sur le calcul de la prosodie à partir de l'analyse syntaxique et qui assure le génie logiciel du synthétiseur.

- et la direction scientifique de l'ensemble du projet : animation, organisation, encadrement des stages d'informatique, convocation et animation des réunions hebdomadaires (petit comité technique) et mensuelles (les 4 partenaires), et coordination générale du projet.

À terme, le synthétiseur intégrera l'analyseur de l'anglais et le diagnostiqueur de langue de Giguet, ce qui permettra de synthétiser du texte bilingue français-anglais en choisissant à bon escient les ressources adéquates (analyse syntaxique, calcul de prosodie, transcription, bases de diphones).

Un tel projet industriel constitue un mode de transmission des résultats de recherche; la mission du chercheur ne se limite pas à chercher, et à trouver, mais aussi à transmettre : transmettre aux étudiants en enseignant et en dirigeant des thèses, transmettre à d'autres chercheurs en publiant, transmettre aussi aux entreprises par la valorisation industrielle des recherches (analyse syntaxique et calcul de la prosodie). Ce projet constitue aussi une validation concrète des concepts sur les liens entre syntaxe et prosodie. Il concrétise mon intérêt croissant pour les formes orales comme faisant partie de mon champ d'étude, avec les formes écrites, ces deux types de formes étant deux instances d'un même objet.

4.3. Valorisation industrielle : direction scientifique du Projet Filtrage de Flux Textuels "Linguix" en collaboration avec Datops

La PME DATOPS[17] (Nimes et Paris) commercialise Pericles et InfoWarner, des logiciels de traitement des flux textuels sur internet (on utilise aussi l'appellation "text mining"). Ces logiciels font de la veille sur des termes choisis par l'utilisateur, et produisent en sortie des séries chronologiques sur les occurrences ainsi que des graphes de cooccurrences de termes. Les flux textuels sont captés sur internet, et sont principalement constitués de dépêches d'agence de presse, et aussi d'articles de presse et de forums de discussion. Les moyens de traitement sont ceux des statistiques lexicales, sans stockage de ressources lexicales.

Le traitement en flux est une problématique particulière et nouvelle : il ne s'agit pas d'interroger un ensemble statique de documents, comme en informatique documentaire classique, mais de prendre connaissance de certaines informations au moment de leur passage dans un flux.

Cette société, avec l'aide du MENRT, a choisi le GREYC pour intégrer dans ses logiciels des outils de traitements linguistiques de l'anglais et du français. Ce projet a démarré en 99, et cette intégration permettra d'affiner l'indexation des termes, mais l'interface utilisateur est inchangée. Les outils de traitement linguistique en cours d'intégration sont les suivants :

• diagnostic automatique de langue : segmenter un flux multilingue en segments monolingues étiquetés (Emmanuel Giguet)

• analyse syntaxique : étiqueter et relier les mots, les syntagmes et les propositions (Jacques Vergne)

• analyse métaphorique : repérer les métaphores, et les évaluer (Stéphane Ferrari)

[17] http://www.datops.com

• analyse structurelle des textes : segmenter un texte en parties différenciées puis reliées (Nadine Lucas)

• un moteur générique écrit en Java (Emmanuel Giguet et Grégoire Cousin) capable de réaliser chacune de ces tâches.

Ce projet permet de développer des composants logiciels portables destinés à s'intégrer d'abord dans les logiciels de la société Datops, et ensuite dans les logiciels d'étude du GREYC ou d'autres laboratoires. En outre, il permet de mettre à l'épreuve des prototypes d'ingénierie linguistique dans un contexte d'utilisation réelle.

En ce qui concerne l'analyse syntaxique, ce projet est l'occasion de développer et tester les ressources linguistiques de l'analyseur de l'anglais, ainsi que l'analyseur bilingue anglais-français muni du diagnostic de langue en entrée, qui permet de sélectionner correctement les ressources linguistiques (ressources lexicales, règles syntaxiques). C'est donc une mise à l'épreuve des méthodes de développement de logiciels multilingues : moteurs indépendants de la langue, ressources déclaratives monolingues ou multilingues.

5. Regards sur le domaine et sur la méthode

5.1. Entre informatique et linguistique, des liens secrets mais forts

Le premier lien entre informatique et linguistique est que dans les deux domaines, on a affaire à des codes : les langages de programmation en informatique, et les langues en linguistique; on a aussi affaire à des opérations sur ces codes, et en particulier des transcodages : la compilation en informatique, et la traduction en linguistique. Mais l'informatique n'échappe pas à la langue, car l'informaticien pense d'abord ses programmes en une langue.

5.1.1. La démarche d'analyse-programmation vue comme une opération de traduction

La démarche d'analyse-programmation peut être vue comme la transmission d'ordres (ou instructions) au processeur, avant l'exécution différée. Si on s'intéresse au code de ces ordres, ils sont conçus en français (ou la langue maternelle de l'informaticien), puis traduits dans le code du processeur (le "langage"-machine), et exécutés par le processeur.

Depuis l'invention des langages de programmation dits "évolués" (FORTRAN puis ALGOL), cette traduction se divise en deux étapes de traduction :

-1- une étape de traduction humaine, du français (une langue, code humain, redondant, implicite, ambigu) vers un "langage" de programmation (un "langage" formel, code intermédiaire, non redondant, totalement explicite, non ambigu),

-2- une étape de **traduction automatique** (par l'interpréteur, traduction imbriquée dans l'exécution, ou par le compilateur, traduction avant l'exécution), de ce "langage" de programmation vers le "langage"-machine du processeur (un "langage" formel aussi, code du processeur, non redondant, totalement explicite, et non ambigu comme le langage de programmation).

Pourquoi cette deuxième étape de traduction est-elle automatisable ? Parce que c'est une traduction entre deux "langages" formels, dont les caractéristiques sont exhaustivement explicitées, car ce sont des artefacts.

Pourquoi la première étape de traduction n'est-elle pas automatisée ? Parce que c'est une traduction d'une langue, dont les caractéristiques sont partiellement inconnues, vers un "langage" formel, dont les caractéristiques sont exhaustivement explicitées. Son automatisation pose des problèmes analogues à la traduction automatique de langue à langue.

5.1.2. Les grammaires formelles entre langue et informatique

Pour un linguiste chomskien, une grammaire formelle sert à modéliser la compétence du locuteur.

Pour un informaticien, une grammaire formelle décrit exhaustivement la syntaxe d'un langage de programmation.

Du point de vue historique, il y a une imbrication entre mathématiques, linguistique, compilation, traduction automatique, avec, au cœur de ce nœud, les grammaires formelles.

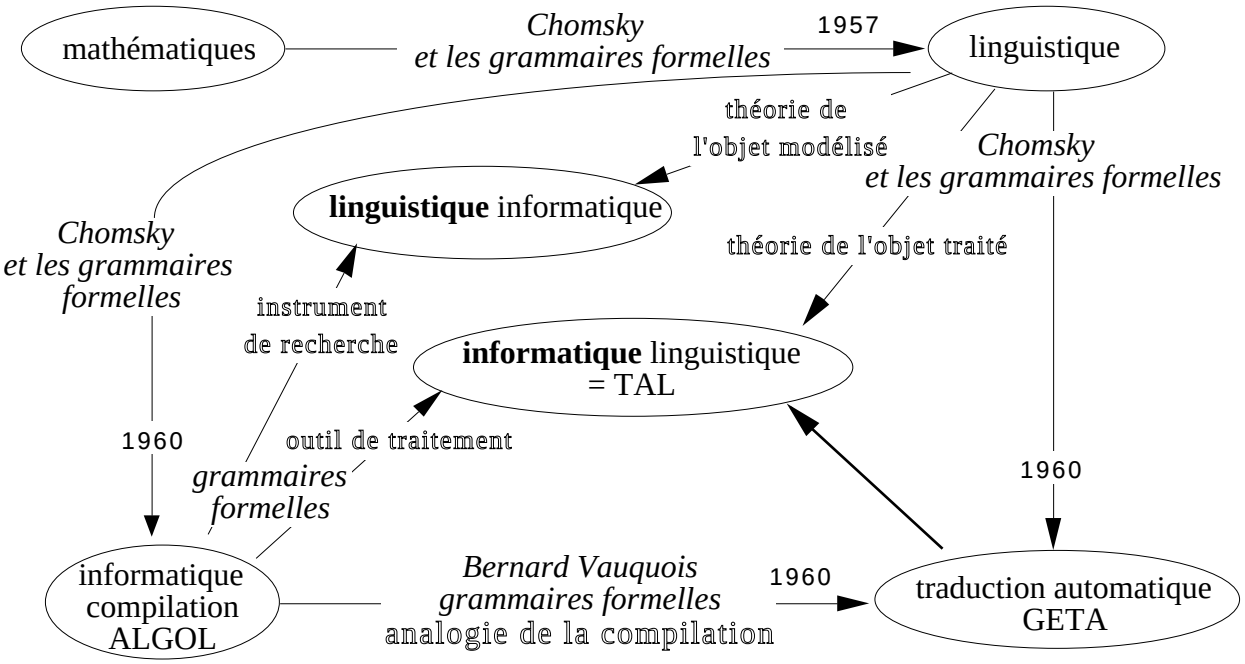
Chomsky a eu une formation initiale en mathématique (ses toutes premières publications sont en mathématique), avant sa formation en linguistique comme élève de Harris; il a travaillé au MIT (55-57) dans un groupe dont un des objectifs était la traduction automatique; il a ainsi importé en linguistique

son bagage mathématique : il crée les grammaires formelles en 1957 (Structures syntaxiques), sur un substrat mathématique, avec un objectif théorique linguistique : modéliser les structures profondes de la compétence du locuteur natif.

ALGOL-60, créé en 1960, est le premier langage de programmation décrit par une grammaire formelle; sa structure de blocs définis récursivement se prêtait bien à l'utilisation de règles récursives.

Dans les années 60, Bernard Vauquois (qui faisait partie du groupe des créateurs d'ALGOL) crée le CETA (puis GETA) à Grenoble, et choisit de fonder la traduction automatique *de langues* de seconde génération sur le modèle de la compilation, traduction automatique *de "langages" formels* (voir pages 2 et 3 de "L'apport scientifique de Bernard Vauquois", par Christian Boitet, dans "Bernard Vauquois et la TAO - Analectes", édité par Christian Boitet, GETA, Grenoble, 1992, et voir page 50 de "TA et TAO à Grenoble... 32 ans déjà", dans t.a.l., revue de l'ATALA, volume 33, numéro 1-2, Klincksieck, Paris, 1992).

Le schéma suivant tente de présenter les pérégrinations des grammaires formelles :



Du point de vue historique, le TAL reçoit donc les grammaires formelles de 3 origines différentes : la syntaxe chomskienne (linguistique), la compilation (informatique), et la traduction automatique (pratique fondatrice du TAL). Les grammaires formelles, créées en linguistique, mais sur une culture mathématique (Chomsky), ont été adoptées avec un grand profit en informatique (langages de programmation et compilation : Vauquois et d'autres), puis ont été transposées (avec moins de bonheur) de nouveau de la compilation vers la traduction automatique (Vauquois), de l'informatique vers le TAL, et donc de nouveau vers la linguistique; ce mouvement de retour au point de départ s'est fait à chaque pas sur des critères surtout opératoires (optimiser la programmation, produire des systèmes de traduction automatique), sans remettre en question les implications linguistiques.

On a fait (et on fait encore) comme si on pouvait traiter une langue de la même manière qu'un langage de programmation, comme s'il n'y avait pas de différences entre langue et code formel. L'analogie langue - code formel a fondé le TAL et la traduction automatique, et maintenant, c'est l'abandon de cette analogie qui permettra de progresser en TAL aussi bien qu'en syntaxe.

5.1.3. Les anglicismes issus du terme anglo-américain "language"

Au cours de leur périple décrit ci-dessus, les grammaires formelles ont emporté dans les pays de langue d'origine latine (surtout France, Espagne, Italie) des termes incluant le mot anglais américain "language", termes (d'origine latine, grâce à Guillaume le Conquérant?) qui ont maintenant submergé par anglicisation les termes de même étymologie latine, en informatique, en TAL, et même en linguistique.

Remarque : la comparaison entre termes doit se faire avec les déterminants, dont le choix ou l'absence change le sens.

1. Avant les grammaires formelles

En français : "une langue", comme le français, l'anglais, est intrinsèquement humaine (sans parler de l'organe charnu); "la langue" est un concept saussurien, opposé à "la parole"; "le langage" est l'abstraction de ce que les langues ont en commun, c'est une faculté spécifiquement humaine; et, pour un linguiste, on ne peut pas dire "un langage", car le langage n'existe qu'unique : "langage" est toujours "le langage" dans le cours de linguistique générale de Ferdinand de Saussure; il existe malgré tout un usage courant de "un langage" comme "une façon de s'exprimer" dans "un langage courant, populaire, relevé, ou académique" (Petit Robert 1).

En anglais (américain) : "a language" regroupe les sens de "une langue" et de "un code artificiel ou animal", et donc a le sens générique de code : toute convention reliant une forme et un sens, pour transmettre ou mémoriser de l'information, d'origine "naturelle" ou artificielle, humaine ou animale; "a natural language" signifie "une langue" : il faut spécifier "language" par "natural" pour signifier "une langue"; "language" seul signifie "le langage", comme le titre de l'ouvrage de Bloomfield (1933).

2. Après les grammaires formelles

En anglais (américain) : une grammaire formelle ("a formal grammar") permet de décrire la syntaxe de "a formal language"; les informaticiens utilisent une grammaire formelle pour décrire la syntaxe de "a programming language"; ils emploient aussi le terme de "a machine language"; "a language" reste générique, et doit être spécifié par "formal", ou "programming", ou "natural"; et "a formal language" signifie "un code artificiel, un code formel", dont la syntaxe peut se décrire par une grammaire formelle.

En français : au début de l'informatique, les informaticiens non-américains lisent des documentations techniques écrites en anglais américain; dans les pays de langue d'origine latine, il est tentant et facile de transposer à l'identique les termes de ces documentations qui sont aussi d'origine latine : "a programming language", "a machine language", "a formal language" deviennent alors : "un langage de programmation", "un langage machine", "un langage formel", comme "a processor" est devenu "un processeur" (mais des néologismes ont été créés : informatique, ordinateur, par exemple); donc "langage" perd son sens spécifiquement humain, pour prendre le sens générique de "a language"; mais cela se passe parmi les informaticiens français, communauté sans contact avec les linguistes français, jusqu'au début de la traduction automatique à Grenoble dans les années 60 : à ce moment, les linguistes français reçoivent la terminologie américaine de plusieurs origines : la linguistique chomskienne, la traduction automatique du GETA, les TAL, et l'informatique française; ils la reçoivent et ils l'acceptent[18] : "langage" perd sa spécificité, même pour des linguistes! Comme les informaticiens, ils emploient "un langage formel" au sujet des grammaires formelles (c'est l'héritage partagé de Chomsky);

[18] Voir le nom donné à la conférence nationale du TAL : TALN, conférence annuelle sur le Traitement Automatique des Langues Naturelles ou sur le Traitement Automatique du Langage Naturel (le sigle est stabilisé, mais pas les termes qu'il recouvre).

mais, pour "une langue", ils disent aussi "une langue naturelle" et "le langage naturel" (qui sont à la fois des pléonasmes et des anglicismes), et même "un langage naturel" (ce n'est pas l'usage courant, qui permettrait "un langage très naturel", mais un anglicisme et pléonasme à partir de "a natural language").

Ce phénomène est étonnant : une communauté professionnelle spécialiste des phénomènes linguistiques accepte l'anglicisation des termes qui désignent l'objet même de sa spécialité; étonnant pour deux raisons : 1) il s'agit des termes centraux de la spécialité; 2) cette spécialité inclut justement les phénomènes d'emprunt d'une langue à une autre.

Remarque : ce problème semble spécifique aux termes partagés entre informatique, TAL et linguistique, car d'autres termes du métalangage anglo-américain des grammaires formelles, qui sont propres à l'aspect linguistique et qui ne sont passés en France que par les linguistes (et pas par les informaticiens), n'ont pas envahi le métalangage français; on ne dit pas par exemple :

"une sentence est composée d'une phrase nominale et d'une phrase verbale",
transposé de : "a sentence is made up of a noun phrase and a verb phrase",
mais : "une phrase est composée d'un syntagme nominal et d'un syntagme verbal".

5.1.4. Comparaison entre une langue et un code formel

Tentons une étude contrastive langue / code formel, en en caractérisant différents critères.

Leurs points communs sont les suivants : ce sont tous deux des codes au sens large (des "languages"), des systèmes de signes conventionnels. Ils ont tous deux : un ensemble de formes possibles (sonores ou visuelles), une syntaxe = règles d'organisation des formes, une sémantique = les correspondances conventionnelles entre les formes et les sens.

Voici une tentative de pointage de leurs principales différences dans le tableau suivant :

critère	langue (natural language)	code formel (formal language)
exemple	français, chinois	"langage" de programmation
origine	sociétés humaines, mères	quelques personnes
utilisation	communication entre humains	un humain commande le processeur
forme	orale, puis écrite	écrite
diachronie	existe : évolution	n'existe pas : figé
lexique	**ouvert**, évolutif	**clos**, figé
morphologie	flexion, dérivation, composition	formes invariables
syntaxe	**peu connue**, évolutive	**totalement explicite**, figée
redondance formelle	grande, inconnue	nulle ou artificielle, connue
récursivité des règles	non (oui selon Chomsky)	oui, sans limite
complexité mémorielle	faible : ≤ 3 sujets en attente	non limitée
corresp.forme/sens	**non biunivoque**	**biunivoque** : 1 forme <-> 1 sens
polysémie, ambiguïté	oui	non
explicitation	incomplète	totale (ex. : anaphore interdite)
métalangage	une langue	une langue aussi

Ces différences sont importantes, sous les trois points de vue suivants :

• Point de vue de l'informatique : ces différences entre langue et code formel permettent d'expliquer la difficulté fondamentale de l'analyse-programmation comme provenant de devoir traduire ce qu'on veut faire faire à l'ordinateur d'une langue dans un code formel : redondance interdite, et explicitation totale obligatoire.

• Point de vue du TAL : plus un traitement porte sur un sous-ensemble restreint (clos et totalement explicite) d'une langue, qui se rapproche d'un code formel, plus le modèle de la compilation est approprié et donne de bons résultats (exemple de TAUM Météo).

• Point de vue de la linguistique : on ne peut expliciter exhaustivement les structures d'une langue par une grammaire formelle, comme on peut le faire pour un "langage" formel.

5.2. Méthode : l'ordinateur, instrument de recherche en syntaxe

Dans cette partie, je tente de préciser les méthodes utilisées dans mes recherches sur la syntaxe des langues, et en particulier le rôle central joué par l'ordinateur, car il fixe la méthode et permet de l'objectiver partiellement en tant que machine à déduire automatiquement[19], ce qui laisse pleinement la part inductive et imaginative de la démarche à l'humain chercheur et expérimentateur.

Je délimiterai d'abord l'objet observable et à théoriser, de manière très restrictive, puis je présenterai la méthode de recherche : comment s'articulent l'observation, l'induction, la formulation des concepts, leur confrontation avec l'objet à l'aide de l'ordinateur, machine à déduire; puis je comparerai cette méthode avec la méthode hypothético-déductive et je tenterai d'expliquer et motiver la place de l'induction. Puis je comparerai rapidement avec l'objet et la méthode de Chomsky.

5.2.1. Délimitation de l'objet observable

5.2.1.1. Objet observable et "objet" construit, réalité et théorie

L'objet observable est concret, réel, éventuellement matériel, et est le point de départ du travail de recherche scientifique, de construction d'une théorie; c'est avec cet objet observable que seront confrontés les concepts. L'"objet" appelé "construit", qui est souvent simplement appelé "objet", est l'aboutissement de la construction théorique, c'est l'hypothèse principale, "le" concept central d'une théorie, ou l'ensemble de ses concepts. Il est couramment admis, surtout en linguistique, qu'il n'y a d'"objet" que construit, que produit par la théorie. Mais quel est alors l'objet observable, réel, confrontable à la théorie? N'y aurait-il pas là une confusion entre délimiter un objet réel, et construire une théorie de cet objet réel, théorie appelée "objet" construit. Posons alors qu'il est possible de délimiter un objet observable **avant** d'en avoir construit une théorie scientifique, même si observer, comme on le sait maintenant depuis Hume et Popper, nécessite déjà des représentations, des "proto-concepts", mais ces représentations ne constituent pas encore une théorie scientifique; de plus le terme "objet" induit une interprétation concrète, alors qu'il est employé là dans un sens abstrait. Dans la suite de ce travail, pour faire une distinction claire entre les deux, j'appelle "objet" l'objet observable, et j'appelle "théorie", "concepts", ou "hypothèse", ce qui est souvent appelé "objet construit".

[19] On pourrait ici l'appeler plus **déducteur** que *ordinateur* ou *calculateur*.

Quand nous observons, nous avons un accès uniquement **indirect** à la réalité qui nous entoure, par l'intermédiaire de nos sens; nous pouvons ainsi construire des **représentations mentales** de cette réalité; on peut appeler ces représentations mentales des "théories", comme Freud a utilisé ce terme à propos des "théories sexuelles infantiles"; ces représentations sont individuelles et/ou collectives, partagées; en fait, un adulte a un accès indirect à la réalité au moyen d'un système perceptif qui utilise, entre autres, ses sens **et** ses représentations mentales acquises. Une théorie scientifique est aussi une représentation mentale d'un objet réel observable, **une** représentation parmi d'autres possibles; mais elle est construite avec des méthodes particulières (telles que celles décrites ci-dessous), et elle est confrontée à son objet observable; elle est explicitée et transmise; de représentation mentale individuelle, elle prend une existence hors des individus qui l'ont créée; elle peut alors subir l'examen de la communauté scientifique du domaine à un moment de l'histoire de ce domaine, et plus tard.

Un objet observable et ses représentations, et en particulier une théorie de cet objet observable, ne sont donc ni identiques, ni assimilables; si l'on croit qu'un objet observable et ses représentations sont identiques, alors on croit aussi qu'une théorie est réelle, et donc intangible ("il faut accepter le réel tel qu'il est"); cette croyance est due en partie à la trop grande discrétion de l'épistémologie et de l'histoire des sciences dans les cursus de formation des chercheurs, et aussi à l'emploi couramment admis du terme "objet" pour une théorie, pour un "objet" construit. Par exemple, l'atome n'est pas un objet réel de la matière, mais un concept sur la matière. C'est justement cette non-identité entre un objet observable et ses représentations qui permet de concevoir une multiplicité de représentations d'un même objet observable, et de remettre en question et abandonner les représentations existantes pour en construire de nouvelles. Une théorie scientifique est contingente à une époque, à un état d'un domaine, aux instruments d'investigation disponibles; elle naît, vit et disparaît, soit intégrée dans une nouvelle théorie plus vaste, soit falsifiée, soit abandonnée au profit d'une nouvelle théorie qui représente mieux l'objet observable.

La théorie, les concepts exposés ici ne sont donc ni un discours vrai, ni une description brute de la réalité, mais plutôt des hypothèses, des questions posées, des conjectures sur des propriétés abstraites des formes des langues, conjectures qui permettent un regard particulier sur les formes des langues, qui sont des "outils à penser", que nous tentons de confirmer sur un petit domaine, et que d'autres falsifieront, ou confirmeront sur des domaines plus vastes.

5.2.1.2. Domaine de validité d'une théorie

Il est souvent dit qu'une théorie devrait exprimer des concepts **universels** sur l'objet observable. Nous proposons ici un objectif moins ambitieux, plus réaliste : l'objectif qu'une théorie exprime des concepts les plus généraux possible sur l'objet observable, circonscrivant ainsi un domaine de validité, qu'on espère être le plus vaste possible, mais pas exhaustif (tout type de texte, pour toute langue passée et présente). Ainsi, ce qu'on appelle d'habitude falsification n'impliquera pas un rejet complet d'une hypothèse, mais seulement une restriction de son domaine de validité; inversement, la confirmation d'un concept sur un autre corpus (autre type de texte, autre langue) impliquera une extension du domaine de validité.

Cette proposition permet de sortir de la vision binaire des énoncés particulier ou universel, et d'introduire un continuum entre les deux. Par définition du domaine de validité, une théorie est valide (corroborée) dans son domaine, et invalide (falsifiée), ou non encore valide (non encore corroborée) hors de son domaine. Une théorie est constituée de concepts sur l'objet observable, de propriétés de cet objet. Ces concepts, ces propriétés sont si possible en petit nombre, généraux et abstraits; leur capacité de représentation est grande. La validité d'un concept sur l'objet observable est évaluée à son adéquation

à le représenter, à sa commodité comme outil mental pour le représenter, et à son efficacité pour agir sur cet objet.

5.2.1.3. Délimitation de l'objet observable

Définir, circonscrire, délimiter l'objet observable est fondamental, et conditionne toute la suite du travail. L'objet observable est ici les formes des langues sous leur "aspect" concret, physique, matériel, et sous leur forme écrite. Leur "aspect", car cet objet d'étude n'est qu'un des aspects sous lesquels on peut étudier une langue. Cet objet est étudié strictement du point de vue des propriétés intrinsèques de ses formes (indépendamment du sens); c'est un objet réel, analogue à un objet de physicien (comme les planètes, les étoiles, les galaxies), dont je tente d'exclure les aspects humains : le sens, la cognition (perception, mémoire), la phonation, la psychologie, la sociologie. L'objet, sous son aspect concret, est un ensemble de textes, tels que les lit un programme d'ordinateur, comme pour l'astrophysicien, pour qui l'objet, sous son aspect concret, est un ensemble de signaux lumineux, d'ondes électromagnétiques, tels que les reçoivent les télescopes et les antennes radio (mais voir ci-dessus l'accès indirect au réel en 5.2.1.1). Mais on rencontre certains aspects humains à leur frontière (surtout les limites de mémorisation), comme trace des contraintes sur les processus de production et réception sur les formes. L'objet ainsi circonscrit peut paraître extrêmement restreint, surtout par rapport aux approches syntaxiques classiques, qui englobent plus ou moins des aspects humains et/ou sémantiques. Cette restriction drastique de l'objet est le prix à payer pour qu'un nouveau pas en syntaxe soit possible; on bénéficie alors de la maturité épistémologique de la physique; on évite l'écueil de l'approche mathématique, détachée d'un objet réel et concret (Chomsky, dont l'"objet" construit - la compétence - est lui-même une hypothèse); on évite l'écueil de l'approche linguistique "cogniticienne" de Chomsky, ou l'approche introspective de Tesnière, qui définissent des objets trop vastes pour lesquels on n'a pas encore de méthodes aussi éprouvées qu'en physique. Pour l'objet observable ainsi délimité, on a des méthodes éprouvées, des instruments nouveaux (dont l'ordinateur), ce qui permet des observations, des investigations, des confrontations concepts - objet impossibles auparavant. En effet, pour rester dans une démarche scientifique, nous sommes contraint de définir un objet pour lequel nous avons des méthodes et des instruments; sinon, nous sommes incapables d'observer l'objet, de confronter les concepts à l'objet : la définition de l'objet observable dans une démarche scientifique à une époque donnée est contingente à l'état des méthodes et des instruments. D'une certaine manière, à cause de ma culture de physicien, la méthode d'analyse utilisée a certains points communs avec la méthode d'analyse distributionnelle (voir les travaux de Hervé Déjean), sans l'avoir choisie explicitement au départ. Une fois ce premier pas réalisé, on pourra bien sûr élargir l'objet, à partir des premiers résultats.

L'objet observable est ici étudié sous sa forme écrite, car c'est sous cette forme que son étude est facilitée aujourd'hui (encore l'état contingent des instruments); mais on gardera présent à l'esprit que l'objet existe sous les deux formes orale (forme première) et écrite, et que les concepts les plus généraux sont à construire pour être valides pour ces deux formes. Quelle est la langue de l'objet? La langue n'est qu'une dimension du choix de corpus, analogue à une autre dimension telle que écrit scientifique - roman, ou oral - écrit, ou niveau de langue; ayons l'ambition que les concepts construits ici puissent être validés sur toute langue, et faisons l'hypothèse que toute langue est soumise aux mêmes contraintes de production et de réception, car toute langue est humaine; ces contraintes impliquent des régularités des formes, régularités indépendantes de la langue. L'objet ainsi délimité est l'objet soumis à observation, point de départ de la théorisation. Les concepts sont construits à partir de cet objet et confrontés à cet objet (voir ci-dessus en 5.2.1.1).

5.2.1.4. Quelques éléments de comparaison avec les "objets construits" de Saussure et de Chomsky

Rappelons maintenant les concepts de Saussure, pour marquer les différences et les restrictions ([Saussure 72], pages 30, 31, 98) :

"la langue" est le **produit social** de la faculté de langage (produit uniquement psychique),

"la parole" est l'**acte individuel** d'utilisation de "la langue" (acte psycho-physique),

"le signifiant" est l'**empreinte psychique** du son, et non pas le son dans sa matérialité.

Saussure crée les concepts de "la langue" et de "la parole", pour définir "le langage" comme l'union des deux, et déclare que son "objet" d'étude est "la langue".

Et rappelons de même les concepts de Chomsky ([Chomsky 71], page 13) :

"la compétence" est la **connaissance** que le locuteur-auditeur a de **sa** langue,

"la performance" est l'**emploi** effectif de **la** langue dans des situations concrètes.

Chomsky crée les concepts de "la compétence" et de "la performance", déclare que son "objet" d'étude est "la compétence".

Ces deux auteurs incluent ainsi dans leurs concepts une part du psychisme humain (Saussure) ou de la connaissance et de l'activité humaines (Chomsky). Dans les deux cas, ils ne définissent pas un objet à observer, étudier, mais ils appellent "objet" un concept purement hypothétique, défini a priori; au départ de leur travail, ils ne définissent pas un point de départ, mais un aboutissement. Je pense important de distinguer l'objet observé, étudié, à théoriser, du résultat de la théorisation, des concepts qui représentent les connaissances construites sur cet objet.

En réaction à la méthode distributionnelle et à la tradition des ses maîtres (Bloomfield et Harris), Chomsky pense qu'on ne peut pas fonder une syntaxe sur l'étude et l'observation de corpus, car un corpus est toujours fini, alors que toute langue rend possible une infinité d'énoncés; cette attitude est surprenante à la lumière des pratiques scientifiques en physique : même s'il est impossible d'observer l'intégralité des étoiles, mais seulement un échantillon, il est possible de bâtir une théorie de l'astrophysique qui modélise comment une étoile apparaît, fonctionne et disparaît; pour étudier un phénomène réel, on n'en étudie toujours qu'une partie infime, à partir de laquelle l'induction est obligatoire; même s'il omet l'induction, c'est justement là un apport important de Popper : toute hypothèse ne peut être que corroborée sur l'échantillon, peut-être falsifiée, mais jamais prouvée; Chomsky lui-même confronte les phrases générées à un échantillon des locuteurs natifs d'une langue; la contrainte de n'étudier qu'un échantillon de l'objet est incontournable quelque soit l'objet.

Chomsky définit son objet comme étant la compétence "du" (des, d'un, de quelques?) locuteur natif; les formes syntaxiques ne l'intéressent pas en elles-mêmes mais seulement en tant que modélisation de cette compétence (c'est un choix surprenant de limiter la modélisation de la compétence aux formes syntaxiques); cet objet est en fait un concept dont la manifestation réelle est la réaction "du" locuteur quand on lui propose une phrase générée; bizarrement, Chomsky n'a pas transposé son objection vis-à-vis de la finitude du corpus vers la finitude du nombre de locuteurs natifs interrogés : quel que soit l'objet observé, l'échantillon réellement observé est toujours fini. Les phrases qu'on trouve dans les ouvrages de Chomsky sont donc des phrases générées et pas des phrases attestées (en fait des propositions), car son objectif n'est pas de rendre compte de phrases attestées.

5.2.2. Méthode de construction de la théorie syntaxique

5.2.2.1. Tentative d'explicitation de la méthode

La **méthode** de construction de la théorie, explicite comment articuler l'observation des corpus, la généralisation des observations en concepts, la confrontation des concepts avec l'objet observable, et le rôle attribué à l'ordinateur et aux programmes. Je tente dans le schéma suivant d'expliciter cette articulation, en commentant chaque étape :

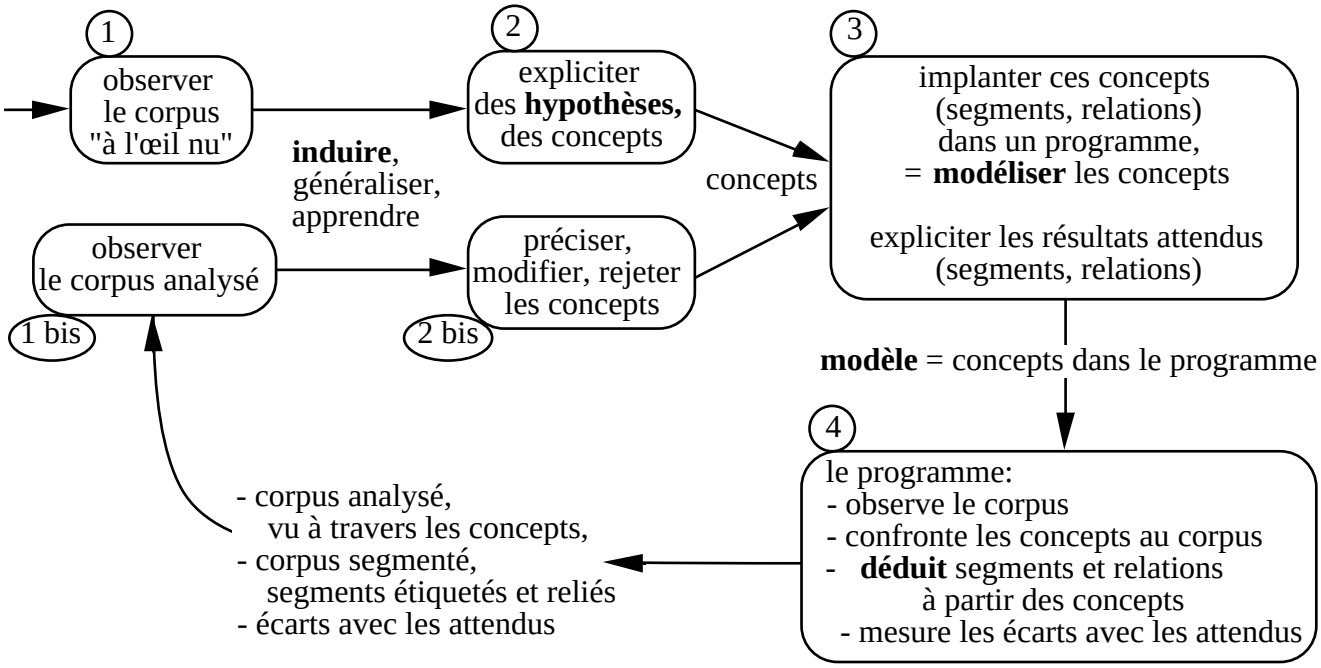


figure 1 : méthode de construction de la théorie syntaxique

1) *Observer le corpus "à l'œil nu"*: le corpus est un ensemble de textes de langues différentes (français, anglais, espagnol). **Mais l'œil n'est jamais nu** : cette observation n'est pas directe, immédiate; elle est dirigée, filtrée, conditionnée par des questions, des connaissances, des concepts déjà acquis sur l'objet; ces connaissances peuvent favoriser l'observation, ou la rendre plus difficile ou même impossible, car pour faire (ou laisser) émerger de nouveaux concepts, il faut parfois renoncer à certains concepts précédents. Il existe une difficulté particulière de l'observation de propriétés jamais observées, car on observe justement par le filtre de concepts, d'attendus. Il est donc normal de ne rien voir de neuf durant les premières observations. Les observations se construisent au fur et à mesure de l'élaboration des concepts. Cette difficulté de l'observation rend difficile la transmission des nouveaux concepts à d'autres chercheurs : il faut qu'ils les acceptent comme hypothèses et nouveaux filtres, nouveaux outils d'observation, comme il faut accepter de regarder dans un télescope pour observer les satellites de Jupiter, puis faire l'hypothèse que les images de ces satellites ne sont pas dues à des défauts du télescope.

2) À partir des observations, *induire, généraliser, apprendre, abstraire,* ce qui permet d'*expliciter des hypothèses*, des concepts hypothétiques, des conjectures. En fait les étapes 1 et 2 font un tout indissociable : observations - concepts. Ces concepts sont principalement des concepts de segments de texte et de relations entre ces segments.

3) *Implanter ces concepts* (explicités, objectivés) *dans un programme* d'ordinateur : ce programme modélise ces concepts, c'est-à-dire décrit des processus sur l'objet d'étude, qui est donc placé en entrée, en analyse, en observation, en réception. Remarquons qu'on a ici une définition pratique de l'objet

d'étude : les textes placés en entrée du programme; l'objet lu par le programme est l'objet à théoriser. Il faut donc définir des processus qui mettent en jeu les concepts hypothétiques : segments et relations. Ces processus consistent à découper l'objet en segments, à identifier ces segments et à relier ces segments. Remarquons que, en plus des concepts implantés dans le programme, il faut définir et expliciter les résultats (segments, relations) bien sûr, mais aussi les algorithmes (les processus) et les informations nécessaires en entrée. L'ordinateur oblige à une discipline d'explicitation complète, et permet à une partie du dispositif expérimental d'être extérieur à l'expérimentateur.

Expliciter les résultats attendus (segments découpés, catégorisés, relations entre segments) : on prend là un deuxième rôle en plus du rôle de chercheur : le rôle d'une personne qui connaît la langue; ce deuxième rôle peut être tenu par une autre personne (c'est le cas pour l'étude de corpus en espagnol).

4) *Le programme* (l'analyseur automatique) exécute les processus suivants :

- observer le corpus à travers le filtre des concepts,
- déduire segments et relations à partir des concepts,
- confronter les concepts au corpus,
- mesurer les écarts avec les attendus.

Le programme est donc un instrument multiple : instrument d'observation (tel un microscope), instrument de déduction régulière, instrument d'expérimentation (dans la confrontation entre concepts et corpus), instrument de mesure (structures et fréquences des segments, propriétés des arbres de dépendance).

Les sorties du programme sont les suivantes :

- corpus analysé, observé à travers le filtre des concepts,
- corpus segmenté, segments étiquetés et reliés,
- statistiques sur les segments et les relations pour l'ensemble du texte,
- écarts avec les attendus.

Les observations, déductions, mesures et statistiques sont là encore complètement fondées sur les concepts.

Les deux étapes suivantes sont parallèles aux étapes 1 et 2, mais l'objet observable est le corpus analysé, passé au filtre des concepts :

1 bis) *Observer le corpus analysé,* qui devient l'observable "à l'œil nu", avec les mêmes difficultés que dans l'étape 1.

2 bis) *Préciser, modifier, rejeter les concepts* en évaluant les résultats, nouvelle phase inductive.

Cette démarche est fondamentalement cyclique. Au cours de multiples cycles, certains concepts s'affinent progressivement, et deviennent de plus en plus adéquats au corpus, et d'autres sont progressivement rejetés. Les cycles sont très nombreux, et l'évolution des concepts très lente; les rejets ou acceptations de concepts ne sont pas opérés sur un seul cycle mais très progressivement sur de nombreux cycles (voir ci-dessous l'évolution des concepts linguistiques en annexe).

En disant : "les concepts deviennent de plus en plus adéquats au corpus", on pose la question de la représentativité du corpus. Le corpus est bien sûr un échantillon restreint de l'objet étudié; et que se passe-t-il quand on change de corpus? Les concepts sont seulement corroborés sur le corpus, jusqu'à falsification éventuelle sur un autre corpus (jusqu'à "preuve du contraire"); le corpus définit un domaine de validité des concepts. Toute démarche scientifique ne rend compte que d'un échantillon restreint de l'objet étudié : une hypothèse n'est jamais prouvée, mais seulement corroborée. Mais plus un concept est fondamental, moins il est local au corpus d'étude, et plus son domaine de validité est vaste : la structure moléculaire de l'eau est la même tant que la taille de l'échantillon est supérieure à une molécule.

5.2.2.2. Discussion sur la méthode de construction de la théorie

Cette méthode se caractérise par la réunion de plusieurs aspects méthodologiques :

- l'utilisation explicite de l'induction (inhérente à la cyclicité de la méthode),
- la formulation de concepts, d'hypothèses, et la déduction à partir de ces concepts : la méthode utilisée intègre la démarche hypothético-déductive,
- le rôle de l'ordinateur comme instrument d'observation, déduction, confrontation, expérimentation,
- la structure itérative, cyclique de la démarche.

L'utilisation explicite de l'induction pose problème aujourd'hui; l'induction a été critiquée, d'abord par Hume, puis par Popper (et ce fut une nécessaire perte d'illusions), mais à tel point que toute induction est souvent gommée des descriptions actuelles d'une démarche scientifique. Bien sûr, l'induction n'est pas un raisonnement, elle n'est pas fondée logiquement, ce n'est pas une méthode totalement explicitable, c'est à la rigueur une démarche mentale, où intuition, créativité, imagination, capacité de synthèse, d'observation, de renoncement aux a priori, jouent des rôles difficiles à cerner; mais elle n'a pas à être comparée à la déduction, ni mesurée à l'aune des caractéristiques de la déduction. Bien sûr, l'œil qui observe n'est jamais nu, car il observe par l'instrument, par le filtre des concepts, des attendus, des représentations préexistants, et si l'induction participe à une explicitation de nouveaux concepts, à une généralisation, ce n'est jamais uniquement à partir des observations, comme si l'observateur était vierge de tous concepts, attendus, ou représentations.

Mais dans une science d'un objet réel, je pense que des phases inductives sont incontournables, et qu'il vaut mieux tenter de les expliciter plutôt que de les exclure arbitrairement de la méthode, sous prétexte qu'on ne peut les fonder logiquement (ce qui est juste), alors qu'elles font intimement partie de cette méthode. Malgré tous les "défauts" de l'induction (en fait seulement ses caractéristiques), il faut expliciter les aspects inductifs de la méthode. On ne peut passer sous silence comment le chercheur apprend sur son objet observable, comment il construit les hypothèses, les concepts. On a pu interpréter la méthode hypothético-déductive comme un apprentissage par essais (les hypothèses) et erreurs (la falsification). Le chercheur fait-il ses essais au hasard? Comment ses erreurs influencent-elles ses essais ultérieurs? D'où viennent les nouveaux concepts, et comment? Ces questions sont importantes; elles font partie de l'explicitation de la démarche, et il ne faut pas reculer devant les difficultés d'y répondre. Les nouveaux concepts ne sont pas créés ex nihilo, comme Mozart composait une sonate. Ils viennent **à la fois** de plusieurs origines : le chercheur induit, apprend, généralise :

- en observant l'objet (observation filtrée par des concepts, des attendus, des représentations préexistants),
- en observant la confrontation des concepts à l'objet,
- en imaginant de nouveaux concepts (imagination baignée dans l'observation de l'objet, et dans les concepts en évolution, ceux de son époque, de sa communauté scientifique, et les siens propres).

Examinons et explicitons l'aspect itératif implicite de la méthode hypothético-déductive, et montrons que, dans cet aspect itératif, le chercheur induit inévitablement à partir de ses observations de la confrontation des concepts avec l'objet. Reprenons le schéma classique de la méthode hypothético-déductive, et explicitons l'itération :

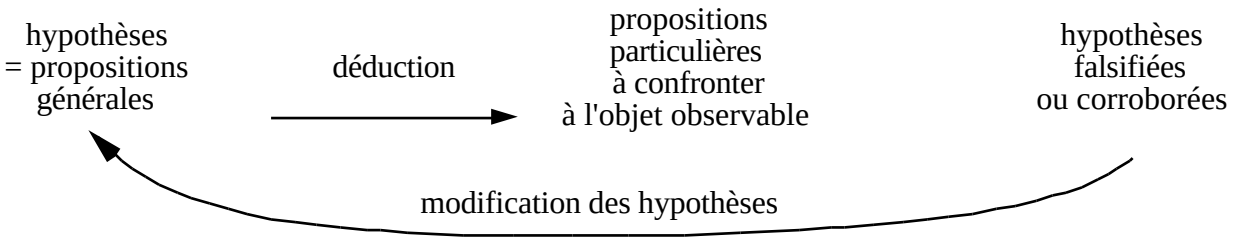


figure 2 : la méthode hypothético-déductive et son caractère itératif

Le chercheur confronte les propositions particulières à l'objet observable, il observe et interprète cette confrontation, qui est elle-même une observation particulière, il diagnostique quelle hypothèse est corroborée ou falsifiée (ce diagnostic est souvent lui-même une nouvelle hypothèse), il décide de modifier certaines hypothèses, et le cycle reprend à partir de ce nouvel ensemble d'hypothèses. Comment fait-il toutes ces opérations, sinon en partie par induction? Y a-t-il une méthode totalement explicitable et fondée logiquement pour observer et interpréter cette confrontation, pour diagnostiquer quelle hypothèse est corroborée ou falsifiée, pour décider de modifier certaines hypothèses, pour décider comment les modifier?

Dans cette itération, on retombe encore sur le problème qu'on voulait évacuer : il y a observation, et il y a induction à partir de cette observation. Comme l'induction est incontournable au cœur même de la méthode hypothético-déductive, réintroduisons-la explicitement dans les deux phases où nous la trouvons :

- comme démarche de modification des concepts à partir de l'observation et l'interprétation de la confrontation des concepts avec l'objet, comme on vient de le voir,

- et comme démarche d'élaboration des concepts à partir de l'observation de l'objet, observation toujours indirecte, médiatisée par des concepts, des attendus, des représentations préexistants.

Ces deux phases n'en font en réalité qu'une seule, comme décrit dans le schéma ci-dessus, dans les étapes 1, 2, 1 bis et 2 bis.

Revenons aux réalités et aux difficultés de l'application de cette démarche dans cette recherche :

- le programme contient un nombre important de concepts, de règles de déduction, d'algorithmes,

- quand il segmente ou relie en faisant des erreurs, il est difficile d'attribuer ces erreurs à un concept, une règle de déduction, un algorithme; quel maillon de cette longue chaîne remettre en cause?

La confrontation entre les concepts et l'objet observable est très difficile à interpréter, une hypothèse n'est jamais corroborée ou falsifiée de manière booléenne, c'est plutôt une tendance dans un sens ou dans l'autre. Dès l'apparition de la lunette astronomique, le problème s'est posé : faut-il attribuer une falsification à un défaut d'une hypothèse, ou à un défaut de l'instrument? À titre d'illustration des difficultés de l'application de cette démarche, voici une métaphore de Popper ([Popper 84], page 111) cité par Chalmers ([Chalmers 87], page 111) :

> "La base empirique de la science objective ne comporte rien d'"absolu". La science ne repose pas sur une base rocheuse. La structure audacieuse de ses théories s'édifie en quelque sorte sur un marécage. Elle est comme une construction bâtie sur pilotis. Les pilotis sont enfoncés dans le marécage mais pas jusqu'à la rencontre de quelque base naturelle ou "donnée" et, lorsque nous cessons d'essayer de les enfoncer davantage, ce n'est pas parce que nous avons atteint un terrain ferme. Nous nous arrêtons, tout simplement parce que nous sommes convaincus qu'ils sont assez solides pour supporter l'édifice, du moins provisoirement."

Pour donner une autre piste de réponse à la question "d'où viennent les concepts?", voici ce qu'écrit Werner Heisenberg ([Heisenberg 72], page 103) :

"Si l'on se pose la question de savoir quel a été le plus grand mérite de Christophe Colomb découvrant l'Amérique, on doit répondre que ce mérite ne consistait pas dans l'idée d'utiliser la forme sphérique de la terre pour atteindre les Indes par la route occidentale; d'autres, avant lui, y avaient déjà songé. [...]. En réalité, le plus difficile, dans ce voyage, c'était certainement la décision de quitter toute terre connue et d'aller si loin vers l'ouest que, avec les approvisionnements disponibles, un retour devenait impossible.

De manière similaire, dans la science, on ne peut gagner une terre nouvelle que si on est prêt, dans une étape décisive, à quitter le terrain sur lequel reposait la science antérieure et à sauter pour ainsi dire dans le vide. C'est ainsi qu'Einstein, en formulant sa théorie de la relativité, avait abandonné cette notion de simultanéité qui faisait partie des fondements de la physique antérieure; ce fut précisément cet abandon du concept de simultanéité qui ne put être accepté par un certain nombre de physiciens et de philosophes, dont certains remarquables, et qui fit de ceux-ci des adversaires acharnés de la théorie de la relativité."

5.2.2.3. Choix de textes composant le corpus

Des textes et non pas des phrases

Il n'y aucun intérêt à isoler un élément de son tout pour l'observer et le théoriser, car on ne sait pas a priori ce qui va jouer dans les phénomènes étudiés; ainsi les corpus étudiés sont des textes entiers et non pas des paquets de phrases extraites de leur texte. Étudier une phrase isolée revient à faire l'hypothèse implicite qu'il n'existe aucune relation de forme entre deux phrases consécutives d'un même texte. Étudier des phrases isolées interdit de faire des études statistiques sur les fréquences des structures, sur les propriétés des arbres de dépendances. L'objet observable d'une étude statistique sur les formes est un texte ou un ensemble cohérent de textes (même auteur, même type, même langue); cette étude statistique indispensable participe à l'affinement, au rejet ou à la validation des concepts, et caractérise le texte, ce qui permet des comparaisons entre textes au sujet de leurs formes.

Ne pas observer des phrases artificielles

Historiquement, il y a eu la pratique chomskienne de générer des phrases artificielles par déduction à partir d'une grammaire, puis de questionner la compétence du locuteur, qui est souvent le linguiste, qui fait donc de l'introspection. Il y a les phrases artificielles de grammaires classiques ou de publications de linguistes, à but pédagogique et explicatif. Mais il y a aussi la pratique de créer des phrases artificielles pour ensuite les étudier comme corpus, avec l'ambition illusoire de bâtir des concepts, d'illustrer des propriétés. On ne fabrique pas des moutons à cinq pattes, pour les observer, quand on veut mieux connaître le mouton des prairies : on se tromperait d'objet observable; c'est une erreur épistémologique courante : on confond objet observable et exemple scolaire ou jeu d'essais pour un programme. Il est préférable d'appliquer un principe d'altérité entre l'observateur et l'objet observé pour éviter l'introspection.

5.2.2.4. Quelques éléments de comparaison avec la méthode chomskienne

Chomsky revendique la démarche hypothético-déductive, dans la pure tradition de Popper. Il transpose ainsi le schéma proposé ci-dessus en 5.2.2.2 :

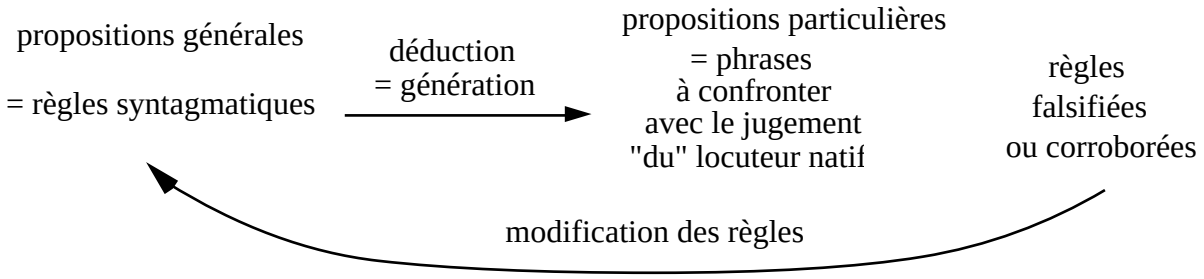


figure 6 : la méthode hypothético-déductive dans la démarche de Chomsky

Les règles syntagmatiques de génération sont à la fois les hypothèses et l'outil de déduction : la génération par réécriture constitue le processus de déduction. Le programme chomskien consiste à construire un système de règles capable de générer toutes (et seulement) les phrases d'une langue (reconnues comme telles par "le" locuteur natif - voir ci-dessus en 5.2.1.5 mon objection sur la finitude du "corpus" de locuteurs natifs). Il n'est pas question d'induction à partir d'un objet observé (on ne sait pas d'où viennent les hypothèses), ni de l'aspect itératif de la démarche. Il n'est pas question non plus d'utilisation de l'ordinateur dans cette démarche.

5.2.2.5. L'introspection chez Lucien Tesnière

Pour Lucien Tesnière, l'introspection est inévitable, car pour lui "la connexion n'a pas de marquant" (voir §12, page 36, et chapitre 17, page 36). L'introspection consiste dans le fait que le linguiste et le "sujet parlant" sont une seule et même personne (voir §13, page 38). Je fais deux objections : si l'on suppose comme il le fait que "la connexion n'a pas de marquant", rien n'empêche de dissocier le linguiste du locuteur en deux personnes différentes, ce qui fut une pratique courante des linguistes; deuxièmement, la connexion est marquée (massivement par la contiguïté) ou calculable à partir d'autres marques, comme le montre le modèle sur ordinateur, qui est capable de calculer les relations sans interroger un "sujet parlant".

6. Synthèse des concepts principaux

En observant le matériau linguistique, on y découvre des traces des processus de production et de réception, par exemple que les segments reliés sont le plus possible contigus, ce qui incite à faire l'hypothèse de la minimisation (et des limites) de l'effort de mémoire au cours des processus.

6.1. Syntaxe : étude et modélisation des processus de production et de réception

6.1.1. Deux visions classiques de la syntaxe

La syntaxe chomskienne est née en linguistique, puis a été importée de manière restrictive en analyse syntaxique automatique, ainsi détournée de sa fonction initiale : de modélisation de la compétence du locuteur natif, elle est devenue modélisation et de fait mode de stockage des structures des phrases à analyser. Il y a donc lieu de parler de la syntaxe chomskienne pour les linguistes, différemment de la syntaxe chomskienne pour les "traiteurs" de langue (3.1.1. et 5.1.2.).

En linguistique, une grammaire syntagmatique est associée à un "générateur - déducteur", et le programme chomskien assigne à ce dispositif dynamique l'objectif de pouvoir produire l'infinité des phrases d'une langue et seulement ces phrases (à soumettre à la compétence du locuteur natif). D'après Chomsky, c'est la récursivité des règles qui permet de modéliser une infinité de structures. Notons que Tesnière, à l'aide de ses concepts, décrit la structure de dépendance d'une phrase donnée, mais il n'a pas d'ambition de décrire a priori la structure de toutes les phrases possibles.

En TAL, une grammaire syntagmatique (ou formelle) est devenue statique, intemporelle; elle a une ambition de double exhaustivité, sur une phrase, et sur une langue : elle explicite **exhaustivement** les structures de tous les constituants d'une phrase donnée, et elle a l'ambition de couverture complète d'une langue, c'est-à-dire d'expliciter **exhaustivement** les structures de toutes les phrases possibles dans cette langue (comme la grammaire d'un langage de programmation permet de compiler tout programme écrit dans ce langage). En TAL, on est resté dans le paradigme de la compilation, analyse des codes fermés, exhaustivement définis.

6.1.2. Déplacement et élargissement du champ de la syntaxe

Dans mes travaux, je propose un élargissement et un déplacement du champ de la syntaxe (voir aussi en 2.0.) :

- un élargissement du champ en incluant dans la théorisation syntaxique les processus de production et de réception de structures;
- un déplacement du champ : des structures vers les processus :
  . en abandonnant l'ambition de description explicite et exhaustive de toutes les structures possibles a priori, mais en décrivant seulement les structures de phrases attestées,
  . en abandonnant les grammaires formelles comme outil de "modélisation" des structures syntaxiques des langues,

. et donc en abandonnant le concept de grammaire en tant qu'inventaire de toutes les structures possibles des phrases d'une langue,
. en décrivant et en modélisant les processus de production et de réception de structures,
. en considérant les structures comme des résultats de contraintes sur ces processus : processus de linéarisation optimisée de l'arbre de dépendance à la production, processus de reconstruction des relations à la réception,
. et en considérant les processus comme des opérateurs en petit nombre opérant sur une infinité d'opérandes, les structures (de manière analogue aux opérateurs de l'arithmétique opérant sur les nombres).

6.1.3. Prosodie et syntaxe

L'écrit et l'oral sont considérés comme deux instances du même matériau linguistique. Je propose donc de les inclure ensemble dans le champ de la syntaxe. Ainsi, dans l'étude de la relation entre prosodie et syntaxe, la question n'est plus de se demander s'il existe une relation entre **la** prosodie et **la** syntaxe, mais de créer **un** modèle qui soit simultanément adéquat aux deux instances, l'écrit et l'oral.

J'ai donc formulé l'hypothèse que la prosodie (hauteur, durée, intensité des phonèmes émis par le locuteur) a pour fonction de permettre à l'auditeur :

1) de segmenter en groupes accentuels (le GA, équivalent à notre SNR) contenant chacun un seul accent primaire (note 2, en 2.2.1.),

2) de reconstruire les relations entre groupes accentuels : 2 GA contigus non reliés sont dits en les séparant par une pause d'autant plus longue que la relation sur la pause est longue; des GA contigus reliés constituent un groupe prosodique (note 5, en 2.2.3.).

Le calcul de la prosodie de la synthèse vocale Kali est conçu sur ces principes, corroborés par les tests perceptifs des utilisateurs (4.2.). Notons qu'une synthèse vocale modélise un processus de réception - production : réception par l'analyseur syntaxique, et production par la synthèse vocale proprement dite.

6.2. Analyse syntaxique automatique non combinatoire

Je propose une autre vision du processus de l'analyse syntaxique, tout en réalisant approximativement les mêmes fonctions (segmenter, identifier les segments et les relier) : au lieu d'avoir en entrée toutes les structures possibles pour toute phrase entrante sous forme d'une grammaire formelle, on a seulement en sortie la structure de la phrase entrante, calculée par application du processus d'analyse (placé en ressource).

*6.2.1. Le processus classique : une recherche d'**appariements***

Le processus classique d'analyse est un processus de reconnaissance, un processus de recherche d'appariements entre la structure d'une phrase entrante et une structure explicitée dans un inventaire supposé exhaustif des structures (la grammaire formelle en ressource); la structure reconnue est préexistante dans la grammaire; cette recherche d'appariements implique un processus combinatoire du fait de la multiplicité des catégories possibles, des règles applicables, ..., avec aucun critère de choix local, mais seulement un diagnostic final : la phrase entrante est ou n'est pas une phrase selon les choix locaux effectués et la grammaire formelle (voir aussi en 3.1.2.). Cet aspect combinatoire est dû à l'ouverture de la langue, alors que la compilation est non combinatoire grâce à la clôture d'un langage de programmation. Si la structure de la phrase entrante n'est pas prévue dans la grammaire, l'appariement

n'est pas possible et aucune analyse n'est produite (c'est l'équivalent de l'erreur de syntaxe en compilation).

La couverture de l'analyseur (domaine des phrases analysables) est directement liée à la couverture de la grammaire formelle (complétude de l'inventaire des structures possibles).

*6.2.2. Un processus de **calcul** de structure*

Le processus d'analyse proposé est un processus de calcul qui produit en sortie la structure syntaxique du texte entrant (c'est-à-dire des segments identifiés et reliés); la structure calculée ne préexiste pas dans les ressources, comme la solution d'une équation ne préexiste pas dans la méthode de résolution; la structure calculée est le résultat de l'application du processus de calcul au texte entrant, comme le résultat d'une équation est le résultat de l'application de la méthode de résolution à l'équation; les ressources ne "modélisent" pas des structures attendues mais seulement le processus de calcul, sous la forme de règles déclaratives interprétées conditions => actions portant sur les valeurs des attributs des segments (voir aussi 3.3.).

Notons que le processus d'analyse modélise un processus de réception.

Ce processus de calcul n'est pas combinatoire, car à aucun moment il ne faut faire un choix aléatoire entre plusieurs voies, avec obligation de les explorer toutes.

Le processus de calcul consiste simplement à passer des règles sur des éléments (caractères, tokens, syntagmes, propositions, phrases, paragraphes,...), c'est-à-dire les appliquer si elles sont applicables, ce qui lui confère une complexité pratique linéaire; plus rien n'empêche alors de traiter en flux à débit constant, sans découper **a priori** une unité à traiter dans sa totalité telle que la phrase classique (unité dont la longueur permet d'évaluer la complexité des algorithmes classiques d'analyse). Un élément du flux est traité complètement, une fois pour toute, en couche unique, avant de passer à l'élément suivant (voir aussi 3.4.).

La couverture de l'analyseur (domaine des phrases analysables) n'est plus liée à un inventaire des structures possibles, mais à la généricité des processus.

6.3. En résumé, évolution :

une langue : fermée et statique	une langue : ouverte et dynamique
modélisation des **structures** avec explicitation totale,	modélisation des **processus** de production et de réception de structures avec explicitation partielle
processus d'analyse syntaxique : une recherche d'**appariements** un processus combinatoire un traitement phrase à phrase	processus d'analyse syntaxique : un **calcul** de structure une complexité pratique linéaire un traitement en flux
en entrée : toutes les structures possibles explicitées dans une grammaire formelle en sortie : les structures de la phrase entrante	en entrée : modèle du processus d'analyse explicité par des règles conditions => actions en sortie : des segments identifiés et reliés

7. Prospective

La prospective de mes travaux de recherche ne peut plus maintenant se concevoir individuellement, mais collectivement dans la constitution progressive du "Groupe Syntaxe" du GREYC.

7.1. Constitution du "Groupe Syntaxe" du GREYC et orientations de recherches

Le "Groupe Syntaxe" du GREYC s'est constitué d'abord par l'encadrement de thésards : Emmanuel Giguet (à partir d'octobre 93), puis Hervé Déjean et Gérald Vannier (à partir d'octobre 95); à partir d'octobre 97, la collaboration avec Nadine Lucas (CR1 SHS, premier CR du GREYC) s'est intensifiée, et son intégration dans le Groupe Syntaxe du GREYC est officielle depuis juin 98, par mutation du LIMSI.

Ma collaboration de longue date avec Nadine Lucas a commencé dès 84 par notre formation au TAL au CERTAL (INALCO), qui nous a transmis la culture de l'École de Prague (Sgall et Hajicova) par l'intermédiaire de Patrice Pognan. L'intégration de Nadine Lucas au Groupe Syntaxe est déterminante : c'est une linguiste japonisante, et ses apports originaux et complémentaires sont sa culture de linguiste, ses recherches sur la syntaxe des textes entiers par études de corpus de langues variées (français, japonais, anglais), et sa pratique de la collaboration avec des informaticiens.

Enfin, Stéphane Ferrari, auparavant en thèse au LIMSI avec Violaine Prince, est en poste à Caen comme MC depuis octobre 98. Sa pratique des travaux sur corpus (détection automatique de métaphores à partir de marques linguistiques), sa participation au projet industriel DATOPS, ainsi que sa culture du codage des documents (SGML, HTML, XML) lui ont facilité son intégration dans le groupe Syntaxe et dans les autres groupes de TAL du GREYC.

La cohérence et l'unité du "Groupe Syntaxe" sont concrétisées par ses options de recherches actuelles et à moyen terme :

- étudier et traiter les formes, sans s'occuper du sens, car c'est ainsi que, dans l'état actuel de nos connaissances, on peut utiliser peu de propriétés formelles très générales, alors que le traitement du sens oblige actuellement à stocker une masse importante de propriétés particulières (ontologiques par exemple);
- prendre une langue comme un ensemble ouvert, évolutif, qu'on ne peut caractériser entièrement; donc ne pas chercher à tout stocker (dictionnaire, grammaire, thesaurus);
- tirer profit de propriétés linguistiques générales avec finesse, traiter le matériau avec peu de ressources très générales, ce qui permet généricité et robustesse;
- travail expérimental sur corpus, à l'aide de l'ordinateur comme outil d'observation, de modélisation et de validation, avec les deux orientations théorique et opératoire;
- approche multilingue : des langues différentes sont des instances différentes du même matériau; sous l'angle du génie logiciel, l'approche multilingue est à la fois une nécessité industrielle et une saine

discipline pour la production de logiciels indépendants de la langue traitée, avec des ressources monolingues extérieures;

- travaux sur l'écrit et sur l'oral, deux instances du même matériau, deux types de formes linguistiques dont les recherches s'enrichissent mutuellement, alors que les communautés de l'écrit et de l'oral sont traditionnellement disjointes;

- sensibilité à l'algorithmique et à la complexité des algorithmes comme signe de la justesse des solutions adoptées;

- traiter en flux, à débit constant (algorithmes de complexité linéaire);

- prendre l'ordinateur comme une machine à calculer des résultats à partir de données, et non pas comme machine à stocker et à choisir parmi des valeurs stockées.

On observe une convergence des concepts : le syntagme non récursif que moi-même et Emmanuel Giguet avons défini se trouve être le groupe accentuel de Gérald Vannier dans le projet synthèse vocale, et est validé par l'approche multilingue de Hervé Déjean; les concepts établis dans le cadre de la phrase sont cohérents avec les travaux de Nadine Lucas sur la syntaxe des textes.

Nos travaux réinvestissent le champ de la syntaxe, champ souvent considéré comme définitivement cultivé, exploité, résolu par les travaux de Noam Chomsky depuis 1957, champ peu fréquenté aujourd'hui, et déserté pour ceux de la sémantique et de la pragmatique. La syntaxe sort de 40 ans de déni de l'intérêt du travail sur corpus, et nous participons au renouveau des travaux sur corpus, au retour au travail sur le matériau, tradition des linguistes européens (Saussure, Benveniste, Tesnière) et américains (Bloomfield, Harris) d'avant 1940. Les recherches en syntaxe à l'aide de l'ordinateur sont un aspect original de notre groupe, car la communauté du TAL (nationale et internationale) est soit encombrée des a priori théoriques chomskiens soit plus préoccupée de retombées opératoires rapides.

Ces orientations permettent à notre groupe d'être innovant et créatif et nous confèrent une avance théorique et opératoire importante dans notre communauté.

7.2. Apports conceptuels des étudiants encadrés

7.2.1. Emmanuel Giguet

Méthode pour l'analyse automatique de structures formelle sur document multilingues

spécialité : informatique, jury : B. Lang (rapporteur), P. Zweigenbaum (rapporteur), M. El-Bèze, J. Mariani, B. Victorri, A. Nicolle (directeur), J. Vergne (directeur), thèse de l'Université de Caen, soutenue en décembre 1998.

- Méthode généralisée et abstraite d'analyse, d'où un processus abstrait et générique d'analyse, paramétrable selon l'unité linguistique construite (mot, syntagme, proposition, phrase), la tâche réalisée (diagnostic de langue, analyse syntaxique) et la langue traitée, ce qui permet de produire du logiciel efficace et léger, car fondé sur ce processus générique et sur les propriétés du matériau linguistique; analyse par repérage de discontinuité, de contraste, de frontières, de différences;

- Le processus abstrait et générique s'instancie dans un moteur générique analogue à un système expert dans lequel les faits sont constitués par le flux entrant, flux sur lequel s'appliquent des règles conditions => actions, à tous niveaux des unités linguistiques;

- Processus capables de manipuler des structures partiellement définies, d'où des processus de calcul à partir de données et non pas des processus faisant des choix parmi des attributs ou des structures

exhaustivement énumérées et explicitées : la machine considérée comme une machine à calculer et non pas comme une machine à stocker et à choisir dans le stock (processus aptes à "connaître" plutôt que "reconnaître");

- Calcul d'une structure formelle en utilisant : (1) des ressources internes (caractéristiques formelles et positionnelles des unités composant la structure à créer), (2) des ressources externes (position et le rôle de la structure dans l'unité linguistique de niveau supérieur qui la contient);

Emmanuel Giguet est en post-doc au GREYC comme chef de projet sur le projet Datops.

7.2.2. Hervé Déjean

Concepts et algorithmes pour la découverte des structures formelles des langues

spécialité : informatique, jury : P. Lafon (rapporteur), F. Debili (rapporteur), D. Bourigault, D. Kayser, K. Zreik (directeur), J. Vergne (directeur), thèse de l'Université de Caen, soutenue en décembre 1998.

- Méthodes de production automatique de structures à partir de corpus bruts de langues variées, et fondation multilingue de concepts linguistiques;
- Méthode générale de constitution automatique ou semi-automatique de ressources linguistiques à partir de textes bruts de langue quelconque (un analyseur prend ses ressources en entrée, alors qu'un système de découverte, à partir du seul corpus, produit des ressources linguistiques en sortie);
- Évolution : passer des ressources exhaustives (le paradigme de la compilation) à des ressources partielles, puis des ressources partielles à aucune ressource sinon un corpus brut;

Hervé Déjean est en post-doc à l'université de Tübigen, sur un projet européen concernant la découverte de structures syntaxiques.

7.2.3. Gérald Vannier

Étude des contributions des structures textuelles et syntaxiques pour la prosodie : application à un système de synthèse vocale à partir du texte

spécialité : informatique, jury : Ph.Martin (rapporteur), Ch.D'Alessandro (rapporteur), Daniel Hirst, B.Victorri (directeur), A.Lacheret (directeur), J.Vergne (directeur), thèse de l'Université de Caen, soutenue en juin 99.

Gérald Vannier a effectué sa thèse dans le cadre du projet Synthèse Vocale (cf. ci-dessus en 4.2). Son rôle a été central et multiple dans ce projet industriel : développement de la majeure partie du logiciel (transposition et adaptation de l'analyseur syntaxique de Vergne, conception et développement du calcul de prosodie), et recherches sur le calcul de la prosodie à partir de l'analyse syntaxique.

Au cours de son étude, il s'est penché plus précisément sur trois manifestations prosodiques importantes : les occurrences et durées des pauses, l'organisation des durées syllabiques et enfin le phénomène de déclinaison de la fréquence fondamentale. Pour chacun de ces points, il a considéré simultanément l'aspect syntaxique et l'aspect organisation textuelle des énoncés comme pistes de recherche pour tenter d'expliquer (et de prédire, dans le contexte de la synthèse vocale) les différentes manifestations.

La réalisation concrète de la synthèse vocale, et le fait que les phrases longues sont très intelligibles apportent une validation perceptive de l'hypothèse que la prosodie permet à l'auditeur de reconstruire les relations (dépendance et coordination) entre les groupes accentuels, par la place et la durée des pauses : place entre 2 groupes contigus non reliés, durée proportionnelle à la longueur des relations.

7.3. Prospective des directions de recherches

Les directions de recherche possibles sont multiples et pléthoriques en rapport avec l'énergie limitée de notre petit groupe. Les résultats actuels doivent être approfondis et il faut continuer à corroborer les hypothèses.

7.3.1. Syntaxe : étude et modélisation des processus de production et de réception

Les recherches sur les processus de production et de réception se feront de plus en plus sur du matériau linguistique écrit et oral, et sur plusieurs langues de front, pour que les concepts produits soient génériques par rapport aux formes écrite et orale, et par rapport à plusieurs langues. L'extension de la synthèse vocale à l'anglais sera un des projets s'inscrivant dans cette évolution.

L'étude de l'optimisation de la linéarisation sera poursuivie sur les formes écrites selon deux directions :

- étude sur corpus, par analyse automatique : repérage des segments optimisés ou non optimisés, étude plus fine des critères d'optimisation, et des métriques des longueurs des relations;

- étude par génération automatique à partir de textes analysés automatiquement, en appliquant automatiquement la linéarisation optimisée : retrouve-t-on le texte analysé, sinon pourquoi ?

Sur des textes écrits, en collaboration avec Nadine Lucas, on recherchera une unification des concepts dans les phrases et au dessus des phrases.

Sur l'oral, le synthétiseur vocal sera utilisé comme outil de recherche pour affiner les hypothèses sur les liens entre linéarisation du réseau des relations et prosodie, par des expériences perceptives.

Toujours sur l'oral, mais par étude du signal de parole sur deux langues (français, anglais), est-il possible de trouver des propriétés générales du groupe accentuel et du groupe prosodique ?

L'hypothèse du moindre effort de mémoire devra être corroborée ou falsifiée expérimentalement en collaboration avec le LPCP (laboratoire de psychologie cognitive de l'université de Caen). On pourra travailler sur les deux processus, production et réception. En production, on peut étudier par exemple la préférence des locuteurs pour les phrases dont la linéarisation est optimisée, et affiner ainsi le concept d'optimisation. En réception, en partant de l'hypothèse sur le processus de mise en relation des syntagmes fondé sur le fonctionnement de mémoires spécialisées par type de relation (mémoriser, puis se souvenir, relier et oublier), on peut étudier la validité psycholinguistique de cette hypothèse en concevant un dispositif expérimental pour corroborer ou falsifier cette hypothèse.

7.3.2. Analyse syntaxique automatique non combinatoire

Les recherches en analyse syntaxique automatique iront vers la consolidation et la généralisation des bases déjà acquises : il s'agira d'abord d'étendre à d'autres langues; l'extension à l'anglais est en cours dans le cadre du projet Linguix en collaboration avec Datops (cf. 4.3.), et les ressources de l'analyse de l'anglais seront incluses dans le synthétiseur vocal; ensuite commencera le travail sur l'espagnol et sur l'allemand, car ce sont les prochaines langues prévues pour le synthétiseur vocal. Ces extensions à d'autres langues permettront d'affiner la méthodologie d'extension à une nouvelle langue.

Le travail sur le calcul des liens de coréférence anaphorique intraphrastique et interphrastique sera repris (j'y avais déjà travaillé en 89-91), et ce calcul sera traité comme un cas particulier de mise en relation par attentes dans des mémoires. L'analyse en flux, qui garde en mémoire les phrases précédentes, facilitera les calculs interphrastiques.

En collaboration avec Nadine Lucas, et à partir des travaux effectués dans le cadre du projet Linguix, on liera plus étroitement analyse de phrase et analyse de texte, en recherchant non pas des processus successifs, mais quasi-simultanés en flux.

La voie de l'analyse syntaxique "sans dictionnaire", explorée durant la thèse, sera reprise avec plus de maturité, car nous sommes toujours persuadés que les ressources lexicales peuvent être encore minimalisées, en abandonnant le mot pour le morphème, en s'inspirant aussi des travaux multilingues de Hervé Déjean. Dans cette optique, on ira vers des ressources lexicales acquises automatiquement sur corpus.

La voie tracée par Hervé Déjean, la constitution semi-automatique des ressources syntaxiques sur corpus, devra aussi être approfondie.

Enfin, l'option du calcul à partir de valeurs par défaut sera approfondie encore plus, en l'accompagnant de la minimisation des ressources lexicales : moins d'éléments et moins d'informations sur ces éléments : uniquement une valeur par défaut, et rien sur les homographies possibles. Cette option est déjà à l'œuvre sur l'anglais où quasiment tout verbe peut être nom.

7.4. Prospective de la valorisation des recherches

Cette valorisation à court et moyen terme se situe à l'intérieur et à l'extérieur du GREYC :

- valorisation et collaboration internes au GREYC : le Groupe Syntaxe offre ses services, ses compétences et sa collaboration aux autres groupes de l'équipe I3, dont les recherches portent sur la sémantique, la pragmatique, le dialogue homme-machine, et le groupe Info-Doc, car ces recherches ont toutes besoin d'un analyseur syntaxique comme module de leurs systèmes. Là aussi, on recherchera non pas des processus successifs, mais quasi-simultanés en flux.

- valorisation et collaboration externes au GREYC : nous allons mettre en place plusieurs directions de valorisation :

. produire et mettre à la disposition de la communauté des corpus étiquetés et des corpus analysés, qui servent de référence et permettent la mise au point et l'évaluation de systèmes syntaxiques,

. offrir un service d'analyse syntaxique automatique du français sur le site internet du GREYC,

. produire (en partenariat industriel dans le projet DATOPS) des ressources modulaires d'analyse syntaxique automatique multilingue (des composants logiciels) intégrables dans des applications plus importantes,

. produire un analyseur de l'anglais (en cours dans le projet DATOPS) pour participer ensuite aux actions d'évaluation internationales et faire connaître ses performances au niveau international.

•

8. Publications

8.1. Publications personnelles :

En résumé :

- la thèse (1989),
- deux articles dans des revues avec comité de lecture (19 pages et 40 pages),
- neuf articles et communications à des conférences d'audience internationale avec actes et comité de lecture,
- trois articles et communications à des conférences d'audience nationale (avec participation internationale) avec actes et comité de lecture,
- 17 communications à des colloques, ateliers, séminaires,
- un rapport interne de 180 pages faisant le point sur mes recherches en 1995,
- trois actions de vulgarisation des recherches : exposés à 2 petits déjeuners de Synergia et démonstrations et explications au public à La Science en Fête en octobre 1996.
- la publication par visualisation graphique des résultats d'analyse (arbres de dépendance) de corpus variés et importants en français, articles de journaux (Le Monde), littérature, textes scientifiques, texte philosophique, sur le site internet du GREYC à l'adresse : http://www.info.unicaen.fr/~giguet.

Thèse

J. Vergne,
"Analyse morpho–syntaxique automatique sans dictionnaire",
spécialité : Intelligence Artificielle et Reconnaissance des Formes, jury : G. Sabah (directeur), D. Kayser (rapporteur), I. Tamba (rapporteur), J.-F. Perrot (président), J.-P. Haton, thèse de doctorat de l'Université Paris 6, 212 pages, 8 juin 1989.

Revues avec comité de lecture

J. Vergne, P. Pagès,
"Symbiose de la syntaxe et de la morphologie dans l'analyse automatique du français avec un minimum de données"
TA Informations (revue de l'ATALA[20]), n°2, pp 27-45 (19 pages), Klincksiek, 1985.

J. Vergne,
"Entre arbre de dépendance et ordre linéaire, les deux processus de transformation : linéarisation, puis reconstruction de l'arbre",

[20] Association pour les Traitements Automatiques des LAngues

Cahiers de Grammaire[21], pages 95 à 136, n° 23, ERSS, Toulouse, décembre 1998.

Conférences d'audience internationale avec actes et comité de lecture

J. Vergne, P. Pagès,
"Synergy of syntax and morphology in automatic parsing of French language with a minimum of data",
CoLing 86 [22], International Conference on Computational Linguistics, pp. 269-271 (3 pages), Bonn, August 1986.

J. Vergne,
"A parser without a dictionary as a tool for research into French syntax",
project note with demonstration, CoLing 90, International Conference on Computational Linguistics, vol. 1, pp. 70-72 (3 pages), Helsinki, août 1990.

J. Vergne,
"Syntax as clipping blocks : structures, algorithms and rules",
communication et démonstration au congrès SEPLN 92 (Sociedad Española para el Procesamiento del Lenguaje Natural), pp. 179-197 and 467 (20 pages), Granada, Espagne, September 1992.

J. Vergne,
"Syntactic properties of natural languages and application to automatic parsing",
communication et démonstration au congrès SEPLN 93 (Sociedad Española para el Procesamiento del Lenguaje Natural), Santiago de Compostela, pp. 182-199 (18 pages), Espagne, septembre 1993.

J. Vergne,
"A non-recursive sentence segmentation, applied to parsing of linear complexity in time",
Communication-démonstration au NeMLaP 94, International Conference on New Methods in Language Processing, Manchester, Royaume Uni, pp. 234-241 (8 pages), Septembre 1994.

E. Giguet, J. Vergne,
"Syntactic Structures of Sentences from Large Corpora",
Demonstration Proceedings of the 5th Conference on Applied Natural Language Processing ANLP'97, Washington, USA, pp. 1-2 (2 pages), April 1997.

E. Giguet, J. Vergne,
"Syntactic analysis of unrestricted French",
In proceedings of the International Conference on Recent Advances in Natural Languages Processing (RANLP'97), pages 276-281 (6 pages), Tzigov Chark, Bulgaria, September 11-13, 1997.

E. Giguet, J. Vergne,
"From Part-of-Speech Tagging to Memory-based Deep Syntactic Analysis",

[21] Les "Cahiers de Grammaire" sont édités par l'Équipe de Recherches en Syntaxe et en Sémantique (Unité de Recherche Associée 1033 du CNRS) de l'Université de Toulouse-Le Mirail. "Cette publication voudrait être le lieu de descriptions détaillées des phénomènes langagiers (aux niveaux syntaxique, sémantique, morphologique ou phonologique) tout en favorisant les recherches et les débats sur les formalismes et les modèles élaborés pour en rendre compte. Les "Cahiers de Grammaire" sont également ouverts aux études pragmatiques et cognitives s'appuyant sur les analyses descriptives ou formelles de la langue. Ils constituent donc un lieu d'échanges et de débats entre **linguistes, psycholinguistes, philosophes du langage, cogniticiens et informaticiens**".

[22] Le CoLing est LA conférence internationale de linguistique informatique, et a lieu tous les deux ans; son taux de sélection est de l'ordre de 25%.

In Proceedings of the International Workshop on Parsing Technologies (IWPT'97)[23], 12 pages, MIT, Boston, Massachussets, USA, September 17-20, 1997.

G. Vannier, A. Lacheret-Dujour, J. Vergne,
"Pauses location and duration calculated with syntactic dependencies and textual considerations for t.t.s. system",
In Proceedings of the 14th International Congress of Phonetic Sciences (ICPhS'99), San Francisco, USA, août 99, 1999

Conférences d'audience nationale (avec participation internationale) avec actes et comité de lecture

J. Vergne,
"Une méthode structurelle de reconnaissance des formes pour l'analyse morpho-syntaxique du français sans dictionnaire",
actes du congrès Reconnaissance des Formes et Intelligence Artificielle (AFCET - INRIA), RFIA'87, Antibes, pp. 933-941 (9 pages), novembre 1987.

J. Vergne,
"Les cadres théoriques des TAL syntaxiques : quelle adéquation linguistique et algorithmique? une étude et une alternative",
avec participation à la table ronde "TALN et linguistique" à TALN'95 [24], Conférence sur le TALN en France, pp. 24-33 (10 pages), Marseille, juin 1995.

J. Vergne, E. Giguet,
"Regards Théoriques sur le “Tagging” ",
Cinquième conférence annuelle : Le Traitement Automatique des Langues Naturelles, TALN'98, pp. 22-31 (10 pages), Paris, France, juin 1998.

Communications à des colloques, ateliers, séminaires

J. Vergne,
"Quelques conseils méthodologiques à des informaticiens pour construire des logiciels qui traitent la langue",
communication au séminaire d'informatique du LAIAC, avril 91.

J. Vergne,
"La syntaxe d'une langue comme em(pile/boite)ment de blocs",
communication au groupe de travail "Traitements Automatiques des Langues" du LAIAC (32 transp.), Caen, juin 1992.

J. Vergne,
"Analyse déterministe sans dictionnaire : bases linguistiques et algorithmes",

[23] IWPT est LA conférence internationale annuelle sur l'analyse syntaxique automatique, organisée par le Special Interest Group on PARSing (SIGPARS) de l'ACL (Association for Computational Linguitics).
[24] Conférence annuelle de la communauté du TAL; taux de sélection : 40% cette année.

communication au groupe de travail "Traitements Automatiques des Langues" du LAIAC (12 transp.), Caen, mai 1993.

J. Vergne,
"D'une recherche théorique à une recherche appliquée : propriétés syntaxiques des langues et application à l'analyse syntaxique automatique",
communication invitée au Colloque de la Maison de la Recherche en Sciences Humaines 93 (10 transp.), Caen, février 1993, et dans "Cognition et Langage", Cahiers de la Maison de la Recherche en Sciences Humaines de l'Université de Caen n°1, 7 pages, juin 1993.

J. Vergne,
"Syntaxe des langues et analyse syntaxique de complexité linéaire",
communication invitée au séminaire du groupe langage et cognition du LIMSI (22 transp.), Orsay, septembre 1993.

J. Vergne,
"Comment qualifier et quantifier la complexité des formes des phrases",
communication invitée au Colloque régional "Illetrisme et Recherche" (19 transp.), Caen, février 1994.

J. Vergne,
"Syntaxe : contiguïté et dépendance - Analyse syntaxique en temps linéaire",
communication au groupe de travail "Traitements Automatiques des Langues" du LAIAC (34 transp.), Caen, mars 1994.

J. Vergne, P. Enjalbert
"Les langages formels: sont-ils adéquats pour modéliser les langues ?",
communication au groupe de travail "Traitements Automatiques des Langues" du GREYC (14 transp.), Caen, mars 1995.

J. Vergne,
"Un concept de syntaxe peut-il avoir un sens du point de vue psycholinguistique? un algorithme d'analyse syntaxique peut-il simuler un processus cognitif?",
communication invitée au séminaire "Modèles et Modélisation" (17 transp.), pôle Sciences Cognitives, Maison de la Recherche en Sciences Humaines, Caen, mars 1995.

J. Vergne,
"Principes d'algorithmique en analyse syntaxique non combinatoire",
communication au groupe de travail "Traitements Automatiques des Langues" du GREYC (14 transp.), Caen, juin 1995.

J. Vergne,
"Une démarche expérimentale en syntaxe des langues : rôle de l'ordinateur?",
communication invitée au séminaire "L'expérimentation en Intelligence Artificielle" (9 transp.), pôle Sciences Cognitives, Maison de la Recherche en Sciences Humaines, Caen, juin 1995.

J. Vergne,
"Analyse syntaxique déterministe : quelques bases linguistiques et psycholinguistiques",

communication invitée au séminaire du groupe langage et cognition du LIMSI (18 transp.), Orsay, avril 1996.

J. Vergne,
"Quelques aperçus sur nos recherches en syntaxe des langues"
communication à la journée du GREYC (18 transp.), septembre 1996.

J. Vergne,
"Des concepts d'algorithmique importés en syntaxe des langues : phrase = arbre linéarisé en minimisant les distances entre nœuds",
communication invitée au séminaire "Algorithmique" du GREYC (21 transp.), Caen, novembre 1996.

J. Vergne, E. Giguet,
"Groupe syntaxe: présentation du groupe syntaxe, quelques nouvelles de l'action GRACE[25], derniers développements de nos recherches",
communication au groupe de travail "Traitements Automatiques des Langues" du GREYC (14 transp.), Caen, avril 1997.

J. Vergne, H. Déjean,
"La technique comme outil des sciences cognitives : l'ordinateur comme outil d'expérimentation sur la langue",
Communication invitée à la journée scientifique de l'ARC[26] (9 transp.), décembre 1997.

J. Vergne,
"Entre arbre de dépendance et ordre linéaire, les deux processus de transformation : linéarisation, puis reconstruction de l'arbre",
Communication invitée à la journée scientifique de l'ATALA[27] (33 transp.) : "Grammaires de dépendance", Paris, mai 1998.

Rapport interne

J. Vergne,
"Esquisse d'une syntaxe des langues concrètes - Application à l'analyse syntaxique automatique",
180 pages, Les cahiers du GREYC, université de Caen, numéro 11, octobre 1995.

Actions de vulgarisation des recherches

J. Vergne,
"Les Traitements Automatiques des Langues à Caen : un pôle régional, une ambition nationale",
exposé au petit déjeuner de Synergia (4 transp.), Caen, 12 avril 96.

J. Vergne,
"Les industriels en binôme avec les chercheurs en Traitements Automatiques des Langues",
exposé et démonstrations au petit déjeuner de Synergia (6 transp.), Caen, 4 octobre 96.

[25] GRACE : **G**rammaires et **R**essources pour les **A**nalyseurs de **C**orpus et leur **E**valuation
[26] Association pour la Recherche en sciences Cognitives
[27] Association pour les Traitements Automatiques des LAngues

J. Vergne, avec la collaboration de E. Giguet, H. Déjean, G. Vannier,
"Analyse grammaticale du français sur ordinateur",
démonstrations et explications au public à La Science en Fête (poster de 22 pages), palais des congrès, Caen, 11 et 12 octobre 1996.

Publication sur internet de résultats d'analyse par visualisation graphique

E. Giguet, J. Vergne,
"Visualisation graphique des résultats d'analyse (arbres de dépendance) de corpus variés et importants en français", sur le site internet du GREYC à l'adresse : http://www.info.unicaen.fr/~giguet, depuis mai 1997.

•

8.2. Autres publications des doctorants encadrés :

Thèses

H. Déjean,
“Concepts et algorithmes pour la découverte des structures formelles des langues”,
spécialité : informatique, jury : P. Lafon (rapporteur), F. Debili (rapporteur), D. Bourigault, D. Kayser, K. Zreik (directeur), J. Vergne (directeur), thèse de l'Université de Caen, 1998.

E. Giguet,
“Méthode pour l'analyse automatique de structures formelle sur document multilingues”,
spécialité : informatique, jury : B. Lang (rapporteur), P. Zweigenbaum (rapporteur), M. El-Bèze, J. Mariani, B. Victorri, A. Nicolle (directeur), J. Vergne (directeur), thèse de l'Université de Caen, 1998.

G. Vannier,
"Étude des contributions des structures textuelles et syntaxiques pour la prosodie : application à un système de synthèse vocale à partir du texte",
spécialité : informatique, jury : Ph.Martin (rapporteur), Ch.D'Alessandro (rapporteur), Daniel Hirst, B.Victorri (directeur), A.Lacheret (directeur), J.Vergne (directeur), thèse de l'Université de Caen, 1999.

Revue avec comité de lecture

H. Déjean,
"Vers une automatisation de l'analyse distributionnelle",
FRACTAL 97, Besançon, 1997. (à paraître dans la revue BULAG)

Conférences d'audience internationale avec actes et comité de lecture

E. Giguet,
"Categorization according to Language : A step toward combining Linguistic Knowledge and Statistic Learning",

Proceedings of the International Workshop of Parsing Technologies, Prague - Karlovy Vary, Czech Republic, 1995.

E. Giguet,
"Multilingual Sentence Categorization According to Language",
Communication à la conférence internationale, from texts to tags : issues in multilingual language analysis, (EACL SIGDAT workshop), Dublin, pp. 76-3-76, Mars 1995.

E. Giguet,
"The Stakes of multilinguality : Multilingual Text Tokenization in Natural Language Diagnosis",
Proceedings of the Pacific Rim International Conference on Artificial Intelligence Workshop,"Future Issues for Multilingual Text Processing", Cairns, Australia, 1996.

E. Giguet,
"Toward an Adequat model for Automatic Syntactic Parsing",
In Poster Proceedings of the Fifteenth International Joint Conference On Artificial Intelligence, IJCAI'97, Nagoya, Japan, August 1997.

H. Déjean,
"Morphemes as Necessary Concept for Structures Discovery from Untagged Corpora",
In Workshop on Paradigms and Grounding in Natural Language Learning, pages 295-299, Adelaide, 1998.

Conférences d'audience nationale (avec participation internationale) avec actes et comité de lecture

H. Déjean,
"Inférences automatiques de contextes distributionnels",
Cinquième conférence annuelle : Le Traitement Automatique des Langues Naturelles (TALN 1998), pp. 142-151 (10 pages), Paris, France, juin 1998.

H. Déjean,
"Quelques marques formelles aidant à une amorce de l'acquisition de la structures des langues",
CAPS'98, Workshop apprentissage et langues, Caen, 1998 (à paraître)

•

9. Bibliographie

9.1. Linguistique informatique et syntaxe

[Biber 93] Douglas **Biber**: *Using register-diversified corpora for general language studies* Computational Linguistics, volume 19, number 2, special issue on using large corpora: II, pages 219 à 242, juin 1993.

[Bloomfield 33] Leonard **Bloomfield**: *Language*, New-York, Allen & Unwin, 1933, trad.fr.: *Le langage*, Paris, Payot, 1970.

[Chomsky 57] Noam **Chomsky**: Structures syntaxiques Point Seuil, 1969, trad.fr.de *Syntactic structures*, La Haye, 1957.

[Chomsky 71] Noam **Chomsky**: *Aspects de la théorie syntaxique*, Seuil, Paris, 1971, traduction en français par Jean-Claude Milner de: *Aspects of the Theory of Syntax*, MIT (Cambridge, USA), 1965.

[Chomsky 80] Noam **Chomsky**, *Essais sur la forme et le sens*, Seuil, Paris, 1980.

[Combettes 88] Bernard **Combettes** and Roberte **Tomassone**: *Le texte informatif, aspects linguistiques* De Boeck Université (Bruxelles) 1988;

[Ducrot 72] Oswald **Ducrot** et Tzvetan **Todorov** *Dictionnaire encyclopédique des sciences du langage*, Point Seuil, 1972.

[Fuchs 87] Catherine **Fuchs**, "L'ambiguïté et la paraphrase en linguistique", pp. 15 à 20, in *L'ambiguïté et la paraphrase*, Caen, Catherine Fuchs (éd.), 1987.

[Fuchs 92] Catherine **Fuchs** et Pierre **Le Goffic** *Les linguistiques contemporaines: repères théoriques*, Paris, Hachette, 1992.

[Fuchs 93] Catherine **Fuchs**, Bernard **Victorri**, Anne **Lacheret**, et al.: *Linguistique et Traitements Automatiques des Langues*, Hachette, 1993.

[Fuchs 94] Catherine **Fuchs** et Bernard **Victorri**: *Continuity in Linguistic Semantics*, Benjamins (Amsterdam), 1994.

[Fuchs 97] Catherine **Fuchs**, "La place du sujet nominal dans les relatives", in *La place du sujet en français contemporain*, pp. 135-178, Duculot, Louvain, 1997.

[Gaifman, 65] Haïm **Gaifman**, "Dependency Systems and Phrase Structure Systems", in *Information and Control* n°8, pp. 304-337, 1965.

[Gosselin 95] Laurent **Gosselin**: *Sémantique de la Temporalité: Un modèle calculatoire et cognitif*, habilitation à diriger des recherches, université de Caen, janvier 1995.

[Grévisse 86] Maurice **Grevisse**: *Le bon usage* douzième édition refondue par André Goosse Duculot (Paris-Gembloux) 1986.

[Grunig 93] Blanche-Noëlle **Grunig**, "Charges mémorielles et prédictions syntaxiques", in *Cahiers de Grammaire* n°18, ERSS, Toulouse, pp. 13-29, 1993.

[Guilbert 75] Louis **Guilbert**: *La créativité lexicale* Larousse (Paris) 1975.

[Hagège 82] Claude **Hagège**: *La structure des langues* Que sais-je? PUF (Paris) 1982.

[Hagège 85] Claude **Hagège**: *L'homme de paroles* Fayard (Paris) 1985.

[Hagège 87] Claude **Hagège**: *Le français et les siècles* Editions Odile Jacob (Paris) 1987.

[Hay 64] D.G. **Hays**, "Dependency theory", in *Language* n°40, pp. 511-525, 1964.
[Kahane 97] Sylvain **Kahane**, "Bubble trees and syntactic representations", in *Proc. 5th Meeting of the Mathematics of Language (MOL5)*, Saarbrücken, pp. 70-76, 1997.
[Kokourek 82] Rostislav **Kokourek**: *La langue française de la technique et de la science* Brandstetter Verlag (Wiesbaden) 1982.
[Larousse 81] Librairie Larousse: *Nouveau dictionnaire des mots croisés Classement inverse* Larousse (Paris) 1981.
[Le Goffic 93] Pierre **Le Goffic**, *Grammaire de la Phrase Française*, Hachette, Paris, 1993.
[Lucas 93a] Nadine **Lucas**: *Syntaxe du paragraphe dans les textes scientifiques en Japonais et en Français* , pages 249-261, actes du colloque international: Parcours linguistiques de discours spécialisés, éditions Moirand et alii., Berne-Paris, pp. 249-261, 1993.
[Lucas 93b] Nadine **Lucas**, Nishina **Kikuko**, Akiba **Tomoyoshi**, K.G. **Suresh**: *Discourse analysis of scientific textbooks in Japanese: a tool for producing automatic summaries* Department of Computer Science, Tôkyô Institute of Technology, Tôkyô, 1993.
[Lucas 95] Nadine **Lucas**: *Le style scientifique en Japonais et en Français* , pages 393-402, Japon Pluriel, actes du 1er colloque de la Société française des études japonaises, éditions Picquier, Arles, 1995.
[Martinet 80] André **Martinet** *Éléments de linguistique générale* Armand Colin 1980
[Mel'cuk 88] Igor **Mel'cuk**, *Dependency syntax : theory and practice*, State University Press NY, Albany, NY, 1988.
[Molière 1670] **Molière**, *Le Bourgeois gentilhomme*, pages 39-48. Presse Pocket. Acte II Scène 4, 1670.
[Portine 92] Henri **Portine**, "Ordre structural et ordre linéaire chez Tesnière", in Actes du Colloque International *Lucien Tesnière aujourd'hui*, CNRS URA 1164, Université de Rouen, pp. 119-127, 1992.
[Saussure 72] Ferdinand de **Saussure**: *Cours de linguistique générale* Payot 1972.
[Stevens 78] Peter **Stevens**, *Les formes dans la nature*, Seuil, Paris, 1978.
[Tesnière 53] Lucien **Tesnière**: *Esquisse d'une syntaxe structurale*, Klincksieck (Paris), 1953
[Tesnière 59] Lucien **Tesnière**: *Éléments de syntaxe structurale*, Klincksieck (Paris), 1982 (1ère édition: 1959)
[Wagner 62] R.L. **Wagner** et J. **Pinchon**: *Grammaire du français classique et moderne* Hachette (Paris) 1962
[Wilmet 86] Marc **Wilmet**: *La détermination nominale* PUF (Paris) 1986
[Zipf 49] George Kingsley **Zipf**, *Human Behavior and the Principle of Least Effort*, Harper, New York, 1949, réédition 1966.

9.2. Analyse syntaxique automatique

[Abeillé 93] Anne **Abeillé**: *Les nouvelles syntaxes*, Colin 1993
[Abeillé et Blache 97] Anne **Abeillé** et Philippe **Blache**, "État de l'art : la syntaxe", in *Traitement automatique des langues*, volume 38, n°2, pp. 69-90, ATALA, Paris, 1997.
[Abney 96] Steven **Abney** (1996), "Part-Of-Speech Tagging and Partial Parsing", in Ken Church and Steve Young and Gerrit Bloothooft, editors, *An Elsnet Book*, Corpus-Based Methods in Language and Speech, Kluwer Academic, Dordrecht.

[Aït-Mokhtar et Chanod 97] Salah **Aït-Mokhtar** et Jean-Pierre **Chanod**, "Incremental finite-state parsing", In Proceedings of the fith Conference on Applied Natural Language Processing (ANLP'97), pages 72-79, Washington, DC USA, April. Association for Computational Linguistics, 1997.

[Andreewsky 74] Alexandre **Andreewsky**, Christian **Fluhr** : *A learning method for natural language processing and application to information retrieval* IFIP Congress août 1974.

[Boitet 92] Christian **Boitet**: *TA et TAO à Grenoble... 32 ans déjà,* t.a.l. (revue de l'ATALA), volume 33, numéro 1-2, Klincksieck (Paris), 1992

[Boitet 92] Christian **Boitet**, dans "Bernard Vauquois et la TAO - Analectes", édité par Christian Boitet, GETA, Grenoble, 1992.

[Brill 95] Eric **Brill**, "Transformation-based error-driven learning and natural language processing : A case study in part of speech tagging", Computational Linguistics, December, 1995.

[Chanod et Tapanainen 95a] Jean-Pierre **Chanod** et Pasi **Tapanainen**, "Creating a tagset, lexicon and guesser for a french tagger", In Proceedings of the European Chapter of the ACL SIGDAT Workshop \From text to tags : Issues in Multilingual Language Analysis", pages 51-57, Dublin, Ireland, March, 1995.

[Chanod et Tapanainen 95b] Jean-Pierre **Chanod** et Pasi **Tapanainen**, "Tagging french & comparing a statistical and a constraint based method", in *Proceedings of the Seventh Conference of EACL (EACL'95)*, pages 149-156, Dublin, Ireland, 1995.

[Chanod et Tapanainen 96] Jean-Pierre **Chanod** et Pasi **Tapanainen**, "A robust finite-state parser for french", In ESSLLI'96 workshop on robust parsing, Prague, Czech, August, 1996.

[Church 93] Kenneth **Church** et Robert **Mercer**: *Introduction to the special issue on using large corpora*, Computational Linguistics, volume 19, number 1, special issue on using large corpora: I, pages 1 à 24, mars 1993.

[Coling 88] Coling 88: *International Conference on Computational Linguistics* (Budapest Hongrie) 1988.

[Collins 96] Michael John **Collins**, "A new statistical parser based on bigram lexical dependencies", In Proceedings of 34th Annual Meeting of the Association for Computational Linguistics (ACL'96), University of California, Santa Cruz, June. Association for Computational Linguistics, 1996.

[Constant 90] Patrick **Constant**, *Analyse Syntaxique par Couches*, thèse de doctorat présentée à l'Ecole Nationale Supérieure des Télécommunications, avril 1990

[Covington 90] Michael A. **Covington**, "A dependency parser for variable-word-order languages", Technical Report AI-1990-01, Artificial Intelligence Programs, The University of Georgia Athens, Georgia 30602 USA, January, 1990.

[Covington 94] Michael A. **Covington**, "Discontinuous dependency parsing of free and fixed word order : Work in progress", Technical Report AI-1994-02, Artificial Intelligence Programs, The University of Georgia, Athens, Georgia 30602 USA, 1994.

[Debili 77] Fathi **Debili** : *Traitements syntaxiques utilisant des matrices de précédence fréquentielles construites automatiquement par apprentissage* thèse de docteur-ingénieur Université de Paris VII septembre 1977

[Debili 82] Fathi **Debili** : *Analyse syntaxico-sémantique fondée sur une acquisition automatique des relations lexicales-sémantiques* thèse de doctorat d'état en sciences informatiques Université de Paris XI 1982

[Debili 82] Fathi **Debili**: *Analyse syntactico-sémantique fondée sur une acquisition automatique de relations lexicales sémantiques*, thèse de doctorat d'état, université Paris 11 Orsay, 1982

[Ejerhed 93] Eva **Ejerhed**: *Nouveaux courants en analyse syntaxique,* t.a.l. (revue de l'ATALA), volume 34, numéro 1, Klincksieck (Paris), 1993

[Feynman 86] Richard P. **Feynman**, Surely you are joking, Mr. Feynman!, Bantam, New York, 1986.
[Fodor 83] J.A. **Fodor**, *The modularity of mind.* Cambridge MA : MIT Press, 1983.
[Giguet 95] Emmanuel **Giguet**: *Multilingual Sentence Categorization According to Language*, communication à la conférence internationale "From texts to tags: issues in multilingual language analysis" (EACL SIGDAT workshop), Dublin, mars 1995.
[Hindle 94] D. **Hindle**, A parser for text corpora. In B.T.S. Atkins, A. Zampolli, (Eds.) *Computational Approaches to the lexicon,* Oxford : Oxford University Press, 103-151, 1994.
[Horn 86] Jean-Paul Horn: *Analyse morphologique du russe Définition d'une automatisation avec un minimum de données Implémentation d'une maquette en Prolog* thèse de doctorat de 3e cycle en traitement automatique des langues Inalco 1986.
[Karlsson 90] Fred **Karlsson**: *Constraint grammar as a framework for parsing running text* CoLing 90 International Conference on Computational Linguistics, vol. 3, pp. 168-173, Helsinki, août 1990.
[López 93] Eduardo **López Gonzalo**: *Estudio de técnicas de procedado lingüistico y acústico para sistemas de conversión texto-voz en español basados en concatenación de unidades* tesis doctoral, E.T.S.I. de Telecomunicación, Universidad Politécnica de Madrid, julio de 1993.
[Marcus 80] M.P. **Marcus**: *A theory of syntactic recognition for natural language*, MIT Press, Cambridge, Mass., U.S.A., 1980.
[Marcus et al. 93] M. **Marcus**, D. **Hindle**, M. **Fleck**, D-theory : talking about talking about trees. *Computational linguistics, 21*,129-136, 1993.
[Marty 85] Fernand **Marty** et R. **Hart** *Computer programs to transcribe French text into speech: problems and suggested solutions* technical report n°LLL-T-6-85, Language learning laboratory, university of Illinois, Urbana, 1985.
[Marty 92] Fernand **Marty** *Trois systèmes informatiques de transcription phonétique et graphémique*, Le Français Moderne, LX n°2, pages 179 à 197, 1992.
[Miclet 84] Laurent **Miclet**: *Méthodes structurelles pour la reconnaissance des formes* Eyrolles (Paris) 1984.
[Miller 56] G. **Miller**, The magical number seven, plus or minus two : Some limits on our capacity for processing information, *PsychologicalReview,* 63, 81-96, 1956..
[Pagès 84] Pascale **Pagès**: *Analyse morphologique automatique du français, extraction des verbes et mise en valeur morpho-sémantique de la dérivation* thèse de doctorat de 3e cycle en traitement automatique des langues Inalco Université de Paris III, 1984.
[Paroubek 98] Patrick **Paroubek**, "Experience in GRACE tagging evaluation", in *FirstInternational Conference on Language Resources and Evaluation proceedings*, Granada, 1998.
[Rady 83] Mohamed **Rady**: *L'ambiguïté du langage naturel est-elle la source du non-déterminisme des procédures de traitement ?*, thèse de doctorat d'état, université de Paris 6, juin 1983.
[Sabah 88] Gérard **Sabah**: *L'intelligence artificielle et le langage*, Hermès, Paris, 2 tomes, 1988, 1989.
[Tapanainen et Järvinen 97] Pasi **Tapanainen** et Timo **Järvinen**, "A non-projective dependency parser", In Proceedings of the fith Conference on Applied Natural Language Processing (ANLP'97), pages 64{71, Washington, DC USA, April. Association for Computational Linguistics, 1997.
[Voutilainen 94] Atro **Voutilainen**: *Three studies of grammar-based surface parsing of unrestricted English text,* Publications of the departement of general linguistics, n°24, université d'Helsinki, 1994.
[Voutilainen et Järvinen 95] Atro **Voutilainen** et Timo **Järvinen**, "Specifying a shallow grammatical representation for parsing purposes", In Proceedings of the Seventh Conference of the European

Chapter of the Association for Computational Linguistics (EACL'95), Dublin, Ireland, March. Association for Computational Linguistics, 1995.
[Voutilainen 95] Atro **Voutilainen**: *A syntax-based part-of-speech analyser*, EACL-95 (7th Conference of the European Chapter of the Association for Computational Linguistics), Dublin, mars 1995.
[Yngve 60] V.H.**Yngve**, A model and an hypothesis for language structure. *Proceedings od the American Philosophical Society*, *104*(5), 444-466 [trad. franç. 1974, Un modèle et une hypothèse pour la structure du langage, In J. Mehler, G. Noizet (Eds.), *Textes pour une psycholinguistique*, Paris : Mouton, 365-420], 1960.
[Weinberg 88] **Weinberg**, Locality principles in syntax and in parsing. *Doctoral dissertation*, Cambridge MA : MIT, 1988.

9.3. Méthode de recherche

[Bachelard 34] Gaston **Bachelard**: *Le nouvel esprit scientifique*, PUF, 1934.
[Bachelard 38] Gaston **Bachelard**: *La formation de l'esprit scientifique*, (1ère éd. 1938) Vrin, 1977.
[Bacon 1620] Francis **Bacon**: *Novum organum*, (1ère éd. 1620 en anglais) PUF, 1986 (trad.fr.).
[Barreau 90] Hervé **Barreau**, L'épistémologie, Que sais-je? n° 1475, PUF, 1990.
[Chalmers 87] Alan **Chalmers** *Qu'est-ce que la science? Popper, Kuhn, Lakatos, Feyerabend* biblio essais Le Livre de Poche, 1987, trad.fr. de *What is this thing called science? An assessment of the nature and status of science and its method*, university of Queensland Press, St Lucia, 1976, 1982.
[Châtelet 92] François **Châtelet:** *Une histoire de la raison*, Points Sciences, Seuil, 1992.
[Einstein 74] Albert **Einstein** et Léopold **Infeld**: *Histoire des idées en physique*, Payot, 1974 (trad.fr.).
[Heisenberg 72] Werner **Heisenberg** *La partie et le tout, le monde de la physique atomique* Albin Michel 1972 (trad.fr.) de "der Teil und das Ganze, Gespräche im Umkreis der Atomphysik", Piper Verlag, München, 1969.
[Kuhn 62] Thomas **Kuhn**: *La structure des révolutions scientifiques*, (1ère éd. 1962 en anglais) Champ Flammarion, 1983 (trad.fr.).
[Popper 84] Karl **Popper** *La logique de la découverte scientifique* (1ère éd. all. 1934, trad.fr. de l'allemand de N. Thyssen-Rutten et Ph. Devaux) Payot, 1984.
[Stevens 78] Peter **Stevens**: *Les formes dans la nature*, Seuil (Paris), 1978.

•