
REPRESENTATION ET CLASSIFICATION EVOLUTIVES POUR LE TRAITEMENT AUTOMATIQUE DU LANGAGE NATUREL

UNE APPROCHE EXPERIMENTALE POUR LE LANGAGE NATUREL AVEC LA PROGRAMMATION A PROTOTYPES

Serge Fleury

ELI (Equipe Linguistique et Informatique) — Ecole Normale Supérieure de Fontenay-St Cloud

31 avenue Lombart - F-92260 Fontenay aux Roses

Mel : fleury@ens-fcl.fr — Toile : <http://www.ens-fcl.fr/~fleury/>

Résumé

Notre travail s'inscrit dans le cadre du Traitement Automatique du Langage Naturel (TALN). Notre démarche vise à la mise en œuvre d'un dispositif informatique cohérent avec les problèmes posés par les phénomènes linguistiques traités par ce dispositif (en particulier la construction du sens). Nous présentons ici le dispositif GASPARG qui construit des représentations des mots sous la forme d'objets informatiques appelés des prototypes ; GASPARG associe à ces objets les comportements syntaxiques et sémantiques des mots en prenant appui sur des informations extraites à partir d'un corpus. GASPARG a pour première tâche de construire progressivement une représentation informatique des mots, sans présumer de leurs descriptions linguistiques ; il doit ensuite reclasser les mots représentés et mettre au jour, de manière inductive, les classes de mots du sous-langage étudié. Nous montrons comment la programmation à prototypes permet de représenter des mots dynamiquement par apprentissage et par affinements successifs. Elle permet ensuite d'amorcer un début de classement de ces mots sur la base de leurs contraintes syntaxico-sémantiques en construisant des hiérarchies locales de comportements partagés. Ce travail met aussi en avant la nécessité de disposer d'un méta-niveau d'analyse dans un dispositif de TALN pour évaluer ou contrôler les opérations réalisées par les processus mis en place dans ce dispositif. Ce méta-niveau d'analyse doit surtout permettre à l'utilisateur du dispositif d'interpréter les résultats construits par les processus de représentation et de classement mis en place.

Introduction

Nous présentons ici la mise en œuvre d'un dispositif expérimental de Traitement Automatique du Langage Naturel qui porte le nom de GASPARG [Fleury 1997]. Ce dispositif vise à établir une représentation et un classement évolutifs de mots représentés sous la forme d'objets informatiques appelés des prototypes. Nous soulignons d'abord les problèmes posés par la représentation d'unités de langue dans un dispositif de TALN. Puis nous présentons le cadre de représentation choisi pour construire des représentations informatiques des mots et des comportements associés. Nous décrivons également les processus mis en place pour une représentation et un classement automatiques des mots. Nous introduisons enfin la nécessité de disposer d'un méta-regard sur les informations manipulées par le dispositif pour évaluer et contrôler les différentes opérations réalisées par le dispositif sur les informations traitées.

1. Une approche expérimentale pour la représentation des mots dans un dispositif de TALN

1.1. Comment représenter la mouvance ?

Le traitement automatique du langage naturel doit passer par une phase de représentation des éléments linguistiques contrainte à figer les informations à modéliser. Or la langue évolue en permanence et les résultats acquis par la description linguistique sont toujours remis en question par la prise en compte de

nouvelles informations. Il en va de même pour les comportements lexicaux : les informations associées aux mots ne sont pas données une fois pour toute. Ces dernières peuvent bouger et remettre en cause des représentations construites à un moment donné. Il n'est donc pas satisfaisant de se contenter d'un modèle apriorique pour construire une représentation des mots. Notre travail de représentation s'inscrit dans une approche expérimentale qui ne présume pas complètement des choses à représenter.

L'hypothèse suivie dans ce travail pour la représentation des informations linguistiques consiste à ne pas prédéterminer de manière figée ni les structures définies pour cette représentation ni leurs classements. Nous convenons avec [Haton 1991] que *"la construction d'une hiérarchie est un processus incrémental"* et qu'une hiérarchie *"évolue et s'améliore en fonction des résultats obtenus jusqu'à ce qu'une certaine forme de stabilité soit atteinte"*. Notre démarche met en place un processus de représentation évolutif : les structures de représentation construites devront pouvoir être affinées dès que de nouvelles informations seront mises au jour. On peut en effet considérer que les comportements des mots ne sont pas tous prédéfinis mais que ceux-ci "émergent"¹ dans le contexte dans lequel ces mots "agissent". Notre démarche consiste en quelque sorte à "faire émerger" les comportements des mots puis à les représenter ou à affiner les représentations existantes et enfin à classer les structures de représentation construites.

¹ Les savoirs généraux que l'on peut associer aux mots ne sont pas toujours pertinents [Biber 1993].

1.2. Observations en corpus des comportements des mots

L'hypothèse suivie ici est qu'il y a peu de sens à vouloir faire de l'acquisition sémantique en dehors d'un sous-langage. Notre travail de représentation et de classement s'appuie sur une recherche initiale au niveau des mots des régularités et des redondances d'utilisation dans des corpus donnés. Le corpus utilisé ici est celui qui est constitué dans le cadre du projet MENELAS [Zweigenbaum 1994] pour la compréhension de textes médicaux. Ce corpus est utilisé par le Groupe de Travail Terminologique et Intelligence Artificielle (PRC-GDR Intelligence Artificielle, CNRS). L'unité thématique de ce corpus a trait aux maladies coronariennes. Les informations extraites sont des arbres d'analyse fournis par des outils d'extraction terminologique (LEXTER², AlethIP³), ces arbres étant ensuite simplifiés dans le but de déterminer les arbres élémentaires de dépendance qu'il est possible d'associer aux mots (ZELLIG⁴) : sont considérés comme élémentaires les arbres mettant en évidence une relation binaire entre deux mots pleins, nom ou adjectif, dans des schémas comme, par exemple, N Prep N ou N Adj. Cette chaîne de traitements est présentée *infra*. Notre travail s'inscrit dans une reprise de l'approche harrisienne [Harris 1970; 1988] et vise à automatiser les traitements de représentation des mots et de leur classement et à souligner les limites de cette induction de savoirs. Un premier objectif est de construire des représentations informatiques évolutives de mots à partir d'informations extraites sur corpus. Le travail de représentation mis en place ne construit donc pas une représentation prédéterminée du sens attaché aux mots ou aux structures syntaxiques représentées, il doit proposer des amorces d'interprétation qui doivent être affinées par un travail d'interprétation plus fin. Le second objectif est de classer les mots et de tendre vers la détermination de classes sémantiques, de manière inductive. Il convient de souligner que le classement automatique présenté ici s'appuie principalement sur des contraintes syntaxiques associés aux mots. Dans notre travail, la syntaxe est utilisée pour dégrossir la représentation et le classement des mots mais ne permet pas à elle seule de classer les mots. A l'inverse des approches harrisienne⁵ et statistiques⁶, notre

approche ne conduit pas à la détermination de classes sémantiques satisfaisantes mais elle constitue une méthode d'amorçage pour l'élaboration de l'ontologie du domaine, nous suivons sur ce point la démarche suivie par [Habert 1996a] : la construction de l'ontologie du domaine étudié nécessite un part d'interprétation. Notre approche de classement des mots est conçue en fait pour aider à accéder aux sens [Habert 1997b]. De plus, à la différence des approches statistiques ou probabilistes appliquées à l'analyse syntaxique qui concentrent l'attention sur les fréquences des mots ou des structures syntaxiques en réduisant l'impact des mots ou des structures à faible fréquence, notre travail de représentation se construit de manière inductive. Les différentes étapes de représentation doivent tenir compte des évolutions potentielles des informations à représenter. Il est donc délicat voire incongru, dans notre cas, de pré-affirmer des poids pour des informations non encore complètement connues. Nous traitons au contraire les informations linguistiques à représenter comme des entités de même niveau de validité, nous nous situons d'ailleurs dans une approche voisine de celle de [Bentch 1995].

1.3. Une approche expérimentale dans un cadre expérimental de représentation

Notre approche vise à travailler à partir d'informations attachées aux mots. Le système doit construire des représentations informatiques de ces mots, il doit ensuite repérer les partages possibles de ces informations. Cette démarche de représentation ascendante crée initialement des macro-structures approximatives qu'il convient d'affiner progressivement, soit en tenant compte de nouvelles informations issues du travail d'extraction à partir de corpus, soit en tenant compte des informations que la situation contextuelle délivre. Il s'agit en fait de mettre en place et d'affiner l'équilibre qui existe entre des savoirs localement répartis et des savoirs partagés qui établissent des liens entre les éléments représentés. Pour mener à bien cette tâche, il est clair que le système doit pouvoir manipuler et assimiler un grand nombre d'informations. Actuellement, le système GASPAS ne prend appui que sur la micro-syntaxe attachée aux mots pour construire un réseau de pôles de comportements partagés par des ensembles de prototypes. Les classes de mots construites sont des ébauches imparfaites qui permettent un organisation du matériel lexical. Ces classes doivent ensuite aider à affiner le travail de description puis de représentation des mots représentés sous la forme de prototypes. De fait la construction progressive de la description des mots passe par un apprentissage manuel de nouveaux savoirs. C'est à l'utilisateur du dispositif d'interpréter et d'évaluer les objets et les résultats produits. Ce travail d'interprétation est d'ailleurs un passage obligé de toutes les approches en classification automatique (en analyse de données par exemple, mais aussi dans des traitements syntaxiques à la Grefenstette [Grefenstette 1993; 1994]). Le système GASPAS s'inscrit d'ailleurs dans une approche qui vise à établir un dialogue entre le dispositif informatique construit et les problèmes posés par les faits linguistiques étudiés. L'intervention manuelle du linguiste s'inscrit parfaitement dans une telle approche. En outre l'environnement de

² LEXTER [Bourigault 1994] est un outil d'acquisition terminologique conçu et réalisé dans un environnement industriel : la Direction des Etudes et Recherches de EDF, pour aider à la mise au point de terminologie.

³ AlethIP est un extracteur de groupes nominaux développé par GSI-ERLI dans le cadre du projet Eureka GRAAL.

⁴ Le logiciel ZELLIG [Habert 1996a; 1996b] est une chaîne de recyclage développée par Benoît Habert et Adeline Nazarenko en C sous Linux et Solaris.

⁵ Dans l'approche suivie par N. Sager, les classes de mots et les formules syntaxiques d'une grammaire de sous-langage sont données comme en correspondance étroite avec les classes d'objets du monde réel et à leurs relations, le travail d'interprétation des résultats construits semble sous-estimé.

⁶ Dans les approches statistiques, les classes de mots sémantiquement pertinentes sont obtenues statistiquement et induites du corpus à partir des seules mesures de similarités entre mots : "En réalité, les listes de mots obtenues ne constituent pas de véritables classes de mots cohérentes et homogènes. Il est souvent difficile de leur attribuer une étiquette globale, et ce d'autant plus que les mesures de similarité constituent des résumés statistiques bruts dont les critères de construction sont effacés" [Habert 1996a].

programmation choisi permet à tout moment d'affiner les représentations construites.

2. PàP : Programmation à Prototypes

L'outil de représentation choisi est la programmation à prototypes (désormais notée PàP) [Blaschneck 1994]. La PàP s'est développée à la fin des années 80, à partir des travaux entre autres de Liebermann et de l'équipe d'informaticiens de Stanford et de Sun Microsystems qui ont par la suite développé le langage à prototypes Self [Self 95]. La PàP conduit à penser différemment pour construire une représentation informatique d'un certain domaine de connaissances. Il ne s'agit pas de partir d'une somme de connaissances figées et connues par avance mais de construire progressivement les entités informatiques suivant les connaissances dont on dispose sur le domaine visé. Si les informations à représenter ne sont pas connues de manière définitive, il est possible de commencer le processus de représentation en utilisant les informations déjà recensées puis d'affiner dynamiquement les objets dès que de nouvelles informations sont disponibles et sans avoir à reconstruire entièrement de nouvelles structures. La flexibilité du modèle de représentation choisi constitue une propriété fondamentale pour développer des processus de représentation capables de rendre compte des aspects mouvants du domaine décrit. Il convient de préciser que la notion de prototype utilisée dans notre travail n'a rien à voir avec celle adoptée par les cognitivistes tels que Rosch [Rosch 1975; 1976]. Par la suite nous utilisons les expressions suivantes. On appelle prototype de mot/d'arbre l'objet informatique (le prototype) défini pour représenter un mot/arbre.

2.1. Une entité = un prototype

A partir d'un élément particulier d'un domaine à représenter, on construit un objet informatique sans passer par un filtre prédéfini (la notion de classe n'existe pas). Si on considère les noms *stenose* et *lesion*, ils partagent des informations et des comportements : ils appartiennent à la même catégorie, ils partagent des comportements syntaxiques (les arbres NPivotPrepN2 et NAdj). Il est donc intéressant d'utiliser la représentation de l'un de ces mots pour représenter l'autre. Si la catégorie syntaxique de *stenose* n'est pas encore représentée, on définit automatiquement les objets pour la représenter avec les attributs adéquats. On construit donc un objet informatique, le prototype *stenose*, à partir d'un savoir (de sens commun) que l'on a sur ce nom : *stenose* est un nom féminin singulier. On crée un objet avec des attributs qui porte ces valeurs.

2.2. Représentation d'une entité similaire par clonage et différenciation

Une fois que l'on a construit un objet particulier, on utilise deux opérations fondamentales pour représenter d'autres éléments sous la forme de prototypes : le clonage et l'ajustement. L'opération de clonage produit une copie conforme de l'objet cloné. Il convient ensuite d'ajuster le prototype issu du clonage pour représenter adéquatement le nouvel élément. Sur chaque objet créé, on peut à tout moment ajuster sa représentation en ajoutant de l'information ou en retirant, les

modifications ne s'appliquant que sur les objets auxquels les messages de modifications sont adressés. Dans notre exemple, l'opération de clonage appliquée au prototype *stenose* permet l'obtention d'une copie conforme de ce prototype. Pour représenter le nom *lesion*, on ajuste ensuite les attributs du prototype résultant en donnant à ces derniers les valeurs adéquates. On peut ensuite représenter les contraintes syntaxiques associés à *stenose*. Si la catégorie de l'arbre élémentaire n'est pas encore représentée, on génère automatiquement un prototype d'arbre pour la représenter avec les attributs adéquats. Puisque les mots *stenose* et *lesion* partagent des arbres élémentaires, les prototypes d'arbres déjà construits pour *stenose* et partagés par *lesion* seront associés au prototype associé à *lesion*. On affecte ensuite aux prototypes de mots construits leurs comportements syntaxiques (les prototypes d'arbres associés).

2.3. Partage des structures et des comportements communs par délégation

La PàP met en place une représentation de savoirs que l'on peut considérer comme peu organisée au départ. Les savoirs représentés sont répartis sur une myriade de structures peu connectées entre elles à priori. La PàP ne rejette pas pour autant toute idée de classification. Elle met en place un système d'héritage simple basé sur la délégation qui permet de factoriser localement des comportements partagés. Dans la mesure où il est possible d'ajuster dynamiquement les prototypes (ajout ou retrait d'attribut), on peut à tout moment construire des liens de délégation. Dans le cas de nos prototypes *stenose* et *lesion*, ces deux objets partagent des savoirs, on factorise donc les comportements communs aux représentations de ces deux mots.

2.4. Self (Sun Microsystems Laboratories)

Self est le langage qui est utilisé ici pour représenter les mots et leurs comportements. Self est un langage qui permet l'héritage multiple et l'héritage dynamique. Il a été conçu en 1986 par David Ungar et Randall Smith [Ungar 1987]. La première implémentation a été réalisée à Stanford en 1987, la dernière version (Self-4.0 [Self 1995]) est disponible depuis juillet 1995. Ce langage est désormais développé par Sun Microsystems avec beaucoup de moyens, ce qui semble indiquer l'importance de ce type de représentation dans le développement de la Programmation à Objets.

Un prototype Self est une entité composée d'attributs. Ces attributs peuvent porter des données, des méthodes ou pointer sur d'autres objets. La définition structurelle d'un objet peut être modifiée à tout moment. On dispose avec Self de primitives qui permettent d'ajouter ou de supprimer dynamiquement des attributs aux objets. Les mécanismes d'ajout(s) ou retrait(s) dynamique(s) d'attributs permettent de construire des représentations par affinements successifs : on peut à tout moment ajouter à un objet de nouveaux comportements ou modifier les comportements de cet objet. Les objets définis dialoguent entre eux via un mécanisme d'envoi de messages.

L'héritage dans la PàP se réalise au travers de la notion de délégation. Pour mettre en place l'héritage avec Self, on commence par créer des objets qui vont porter les comportements partagés. On établit ensuite un lien entre ces objets et les prototypes qui délèguent les comportements factorisés en inscrivant dans ces prototypes le chemin de délégation défini. Pour inscrire un chemin de délégation dans un objet on ajoute un attribut `parent` (nom de l'attribut suivi d'une étoile) qui pointe sur l'objet qui porte les comportements délégués. Dans le langage Self, un parent commun à plusieurs prototypes est appelé un objet `traits`.

2.5. Le travail sur les gros corpus et les limites des machines qui portent Self

Les résultats produits par le dispositif GASPARG sont en fait limités actuellement par les contraintes matérielles imposées par ce langage expérimental sur les machines que nous utilisons. Nous sommes contraint actuellement de restreindre considérablement les informations à traiter par GASPARG (partie 6 *infra*). Il faut en effet beaucoup de mémoire et d'espace disque sur les machines qui portent le système Self pour mettre en œuvre cette représentation de la mouvance. Cette incapacité actuelle du système Self à assimiler la génération d'un grand nombre d'objets met surtout en avant les problèmes complexes liés à la mise en œuvre de ce type de modèle de représentations évolutives. Cette modélisation de la mouvance dans les outils de représentation coûte chère en ressources pour les systèmes sous-jacents. Le dispositif GASPARG peut être considéré comme une expérience pilote qui offre une image partielle, pour le moment, des traitements réalisés et des résultats à venir.

3. Acquisition de savoirs en corpus

La phase d'acquisition d'informations à partir de corpus prend appui sur la systématisme structurelle et sémantique propre aux sous-langages [Harris 1970; 1988] afin de mettre au jour les proximités de cooccurrences entre mots pour dégager les relations sémantiques sous-jacentes. Le point de départ du travail de représentation est donc constitué par le corpus MENELAS. Les informations utilisées par les processus de représentation informatique des mots et des arbres associés sont issues d'une chaîne de traitements composée des logiciels LEXTER, AlethIP et ZELLIG. Le but de ces outils est d'une part d'extraire des informations à partir de corpus (LEXTER, AlethIP) et d'autre part de simplifier ces informations puis de caractériser leurs fonctionnements (ZELLIG). LEXTER prend en entrée des textes arbitrairement longs et produit des arbres d'analyse de séquences nominales en décomposant ces séquences en Tête et Expansion et ce de manière récursive. Il est important de souligner que la démarche de LEXTER est endogène c'est à dire que le travail d'analyse s'appuie uniquement sur les résultats d'analyse déjà construits pour analyser des séquences ambiguës. Si le corpus contient la séquence "angine de poitrine instable", on a deux analyses possibles pour cette séquence :

- (1) [angine de poitrine] instable
- (2) angine de [poitrine instable]

LEXTER va prendre appui sur des séquences déjà analysées pour produire une analyse de cette séquence. Si on a la séquence "angine de poitrine" et si l'on n'a pas "poitrine instable", LEXTER produira l'analyse (1). L'approche endogène suivie par LEXTER s'inscrit parfaitement dans le cadre de représentation qui est le nôtre. Le logiciel ZELLIG a ensuite pour tâche de simplifier les arbres d'analyse fournis ici par LEXTER et de mettre en évidence les relations élémentaires de dépendance entre mots pleins, nom ou adjectif, dans des schémas comme, par exemple, N Prep N ou N Adj. Les arbres issus de LEXTER sont d'abord transformés en arbres syntagmatiques via le transducteur FRT (un module de ZELLIG) [Habert 97a]. Le programme Cyclade (un autre module de ZELLIG) est ensuite chargé de déterminer les arbres élémentaires via un filtrage de quasi-arbres [Habert 96b]. Ces dépendances élémentaires associent à un élément gouverneur (nommé `tête`) soit un argument soit un `circonstant`. On résume ici de manière synthétique ce que la chaîne de traitements réalise pour produire les arbres élémentaires à partir du corpus initial. Le point de départ est constitué de rapports médicaux. LEXTER produit des arbres d'analyse de séquences nominales en décomposant ces séquences et en les structurant. Sur la séquence "alteration severe de la fonction ventriculaire gauche", on obtient l'analyse suivante :

```
[T [T alteration] [E severe] [E de la
 [T [T fonction] [E ventriculaire]
 [E gauche]]]
```

ZELLIG met en évidence les dépendances élémentaires dans ces arbres d'analyse. A partir de l'analyse précédente, les arbres élémentaires (a, b, c, d) sont mis au jour :

- (a) alteration severe,
- (b) alteration de fonction,
- (c) fonction gauche,
- (d) fonction ventriculaire.

4. Du corpus jaillit un réseau de prototypes

4.1. Des observations en corpus aux prototypes

GASPARG dispose, au départ, d'informations extraites à partir d'un corpus (sous la forme d'un fichier texte) : pour chaque mot, GASPARG dispose d'informations morphologiques et sémantiques décrivant ces mots, d'une liste d'arbres élémentaires et d'une liste d'arbres d'analyse associés aux arbres élémentaires (des contraintes peuvent être attachées aux composants des arbres, on y reviendra *infra*). Ces descriptions sont généralement peu précises ou en tout cas sous-déterminées. GASPARG utilise ces informations pour construire des prototypes afin de représenter les mots et leurs comportements (les arbres associés). Le processus de représentation s'appuie uniquement sur ces informations pour construire des prototypes.

4.2. Représentation automatique des mots

Si le mot à représenter possède une représentation prototypique, GASPARG conserve l'objet trouvé. Si le mot à représenter ne possède pas de représentation prototypique, et s'il n'existe aucune représentation prototypique de sa famille catégorielle, GASPARG commence par créer de toutes pièces une

représentation prototypique de cette nouvelle famille catégorielle, puis il construit un prototype de ce nouveau représentant de cette famille (en tenant compte des informations fournies pour décrire ce nouvel élément).

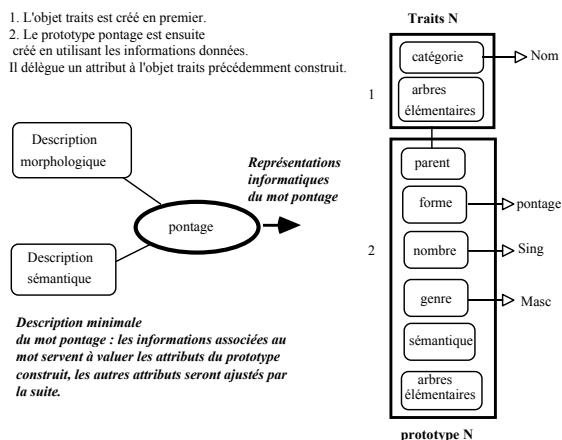


Figure 1. Génération automatique d'un prototype de mot

Si le mot à représenter ne possède pas de représentation prototypique et s'il existe déjà un prototype de la même famille catégorielle, GASPAR utilise les opérations de clonage et d'ajustement pour représenter ce nouvel élément.

4.3. Représentation automatique des arbres associés aux mots

Le dispositif GASPAR procède de la même manière pour la représentation des arbres. Avant de représenter ces arbres élémentaires, le dispositif GASPAR vérifie si ces arbres disposent déjà d'une représentation prototypique. Si elle n'existe pas, il la crée automatiquement en tenant compte des informations fournies : constituants et contraintes⁷.

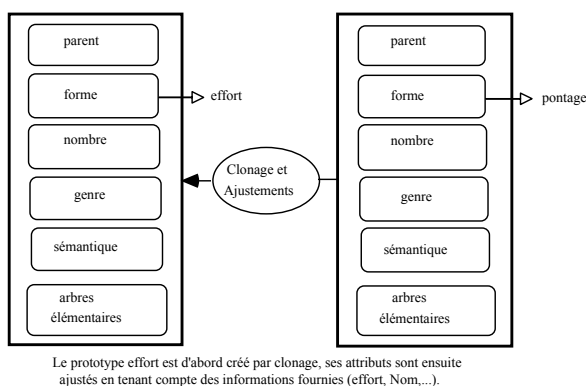


Figure 2. Génération d'un prototype de mot par clonage et ajustement.

Dans la figure ci dessous (figure 3), le dispositif GASPAR construit les structures pour représenter les arbres associés à pontage en tenant compte des

informations données pour la description de ces arbres. Puisque les mots pontage et effort partagent des arbres élémentaires (les arbres N1PrepNPivot et NAdj), les prototypes d'arbres déjà construits pour pontage et partagés par effort seront associés au prototype de mot associé à effort.

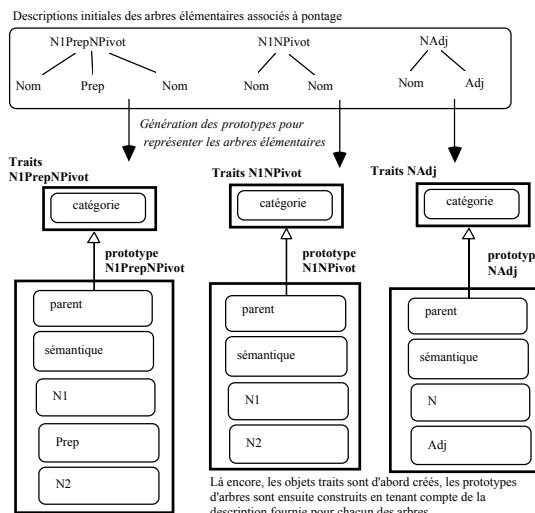


Figure 3. Génération de prototypes d'arbre.

Pour représenter les arbres d'analyse, le dispositif GASPAR vérifie là encore si ces arbres disposent déjà d'une représentation prototypique. Si elle n'existe pas, il la crée automatiquement en tenant compte des informations fournies. Le dispositif GASPAR affecte ensuite les prototypes d'arbres aux prototypes de mots auxquels ils sont associés. De même, il associe les prototypes d'arbres d'analyse construits aux prototypes d'arbres élémentaires associés. Pour chaque mot, le dispositif GASPAR a donc construit un mini-réseau de prototypes décrivant la micro-syntaxe associée à ce mot. La représentation des mots reste bien évidemment sous-déterminée. Si de nouvelles informations sont disponibles, on peut affiner la représentation des mots et des arbres en utilisant le potentiel dynamique de Self. Dès cette phase de représentation, l'utilisateur peut intervenir pour ajuster les objets suivant les nouvelles informations disponibles.

5. Le réseau de prototypes dévoile une hiérarchie évolutive

5.1. Construction d'un réseau de hiérarchies locales de comportements partagés

GASPAR amorce un classement des prototypes de mots en fonction de leurs comportements (les arbres associés aux mots). Il recherche donc les comportements⁸ partagés par les prototypes de mots. Si on considère les noms *stenose* et *lesion*, ils partagent des comportements (les arbres

⁷ Les contraintes associées à un arbre décrivent par exemple des informations qui filtrent ou sélectionnent les constituants possibles pour cet arbre.

⁸ On classe donc les objets représentant des mots sur la base des comportements réels associés aux mots ainsi modélisés. Les comportements informatiques qui ne décrivent pas les mots dans leur activité langagière ne sont pas pertinents pour ce classement.

NPivotPrepN2 et NAdj). Si on considère maintenant le nom *angioplastie*, celui-ci entre dans des constructions du type "indication de angioplastie" (l'arbre N1PrepNPivot). GASPAR construit donc un pôle de comportements partagés qui va porter les prototypes d'arbres élémentaires communs. Il établit aussi un lien de délégation entre ce pôle et les prototypes représentant les mots *stenose* et *lesion*. Sur un ensemble plus important de prototypes de mots de même catégorie, GASPAR met en place, automatiquement, un réseau de pôles de comportements partagés en définissant des hiérarchies locales sur des sous-ensembles de prototypes de mots. Ce premier classement s'appuie sur les comportements syntaxiques attachés aux prototypes de mots. Il ne dit rien de plus sur les agrégats de comportements partagés construits.

5.2. La classification évolutive est en marche

Les processus construits permettent en fait d'évaluer plusieurs types de recherches de comportements partagés sur les objets.

- (1) Le dispositif GASPAR peut tout d'abord rechercher sur tous les mots d'une même catégorie s'il existe des arbres élémentaires en commun. Si tous les prototypes de mots d'une même catégorie partagent exactement les mêmes prototypes d'arbres élémentaires, l'objet *traits* qui porte les comportements partagés de cette catégorie est mis à jour : il portera ces comportements communs. Dans tous les cas, les prototypes de mots portent, quant à eux, leurs comportements propres.
- (2) Le dispositif GASPAR peut ensuite rechercher sur les prototypes pris deux à deux s'ils partagent des arbres élémentaires. Si deux prototypes de mots d'une même catégorie partagent un ou plusieurs prototypes d'arbres élémentaires, un objet *traits* est automatiquement construit pour porter ces comportements partagés. Dans ce cas, le dispositif GASPAR ajoute automatiquement aux prototypes concernés un attribut *parent* qui pointe sur ce nouvel objet porteur de comportements partagés. Il est possible de réaliser ce type de recherche sur deux mots particuliers ou sur l'ensemble des mots (pris deux à deux et dans chaque catégorie).
- (3) Le dispositif GASPAR peut aussi évaluer les comportements partagés sur des sous-familles de prototypes de mots de même catégorie. Si plusieurs prototypes de mots d'une même catégorie partagent exactement les mêmes prototypes d'arbres élémentaires, un objet *traits* est automatiquement construit pour porter ces comportements partagés. Dans ce cas, le dispositif GASPAR ajoute automatiquement aux prototypes concernés un attribut *parent* qui pointe sur ce nouvel objet porteur de comportements partagés.
- (4) Le dispositif permet enfin d'évaluer automatiquement les différences comportementales des arbres élémentaires. Il est en effet possible d'établir une recherche sur les arbres élémentaires de même catégorie des comportements partagés (arbres d'analyse) par ces arbres élémentaires. Ce classement utilise une démarche similaire à celle qui est utilisée pour classer les mots. Si plusieurs

prototypes d'arbres élémentaires d'une même catégorie partagent exactement les mêmes prototypes d'arbres d'analyse, un objet *traits* est automatiquement construit pour porter ces comportements partagés. Là encore, le dispositif GASPAR ajoute automatiquement aux prototypes concernés un attribut *parent* qui pointe sur ce nouvel objet porteur de comportements partagés. GASPAR permet donc d'activer des processus de classement qui proposent des regards multiples et croisés sur les informations représentées. Ces processus construisent des réseaux de hiérarchies locales entre prototypes de mots et prototypes d'arbres ou entre prototypes d'arbres, ces liens multiples constituent autant de pistes de sens à interpréter.

5.3. Une amorce de classement guidée par la syntaxe

Les comportements associés aux représentations prototypiques des mots sont de simples squelettes syntaxiques. Il est clair que la simple recherche de similitude de comportements syntaxiques entre mots n'est pas suffisante pour une classification cohérente des mots représentés. La syntaxe dégrossit en quelque sorte la représentation mais ne permet pas à elle seule de classer les mots ; la syntaxe est incapable à elle seule de délimiter des classes de mots reflétant une notion. On peut avoir un rapprochement de certains mots suivant certains comportements syntaxiques, mais aucune classification ne se révèle directement à partir des comportements syntaxiques [Habert 1996a]. Repérer des similitudes formelles entre les représentations créées est une tâche à la portée de la programmation à prototypes et le classement obtenu ici est une amorce de classement sur lequel un travail d'interprétation reste à faire. Évaluer ces similitudes formelles et interpréter les regroupements restent des tâches auxquelles doit être confronté le linguiste dans la mesure où il est le seul à pouvoir y apporter une réponse précise.

5.4. Une démarche expérimentale et en spirale

Notre démarche cherche à réaliser une adéquation entre les occurrences linguistiques réalisées et les prédictions de représentations construites. Il ne s'agit pas de produire d'emblée un résultat définitif qui réalise cette adéquation de manière parfaite ; mais plutôt de tendre vers cette adéquation, par touches successives, en affinant les prédictions construites. La mise en œuvre des représentations est donc conçue comme un mécanisme évolutif qui, d'une part, doit tenir compte d'un nombre important de sources de connaissances, et d'autre part, doit être capable d'intégrer de nouvelles informations à chaque étape. GASPAR peut disposer en amont de plusieurs couches d'informations. Si le travail d'extraction de savoirs est capable de mettre en évidence différents types d'informations pour décrire un mot, GASPAR utilise ces informations dès la première phase de génération du prototype construit pour représenter ce mot. On peut aussi envisager de procéder à une acquisition de savoirs en réitérant la phase initiale de recherche d'informations à partir du corpus. On pourrait ainsi moduler et modulariser cette phase d'acquisition de savoirs en activant différents flux

de savoirs. Si les informations attachées aux mots ne sont pas disponibles dès la première phase de génération des prototypes, il sera toujours possible d'ajuster les représentations en utilisant un nouveau flux d'informations disponibles ultérieurement. Il est aussi possible de projeter les résultats sur des savoirs établis par ailleurs et d'utiliser les résultats de ces projections pour ajuster les représentations. Le processus de représentation peut donc se développer en réitérant les phases suivantes : représentations à partir de savoirs extraits d'un corpus, projections des résultats, ajustements des représentations, ajustements du classement. Ces différentes étapes peuvent induire des phases intermédiaires de travail manuel pour corriger les états de représentation produits [Mikheev 94]. Si GASPAR peut automatiser le classement des prototypes de mots sur la base des comportements qui leurs sont associés, les résultats restent à qualifier, à nommer : dans notre dispositif, c'est l'observateur conscient qui donne le sens. C'est en examinant à la main les objets et les rapprochements constatés que l'on pourra leur donner un nom c'est à dire nommer les choses. Il s'agit en fait de tendre vers une cohérence des classes sémantiques issues des processus de classement afin de dégager par affinements successifs des descriptions sémantiques pour les mots.

6. Résultats construits

Deux corpus de travail sont utilisés pour mettre en œuvre les outils définis : le premier est obtenu via le travail d'extraction réalisé par LEXTER, le second via l'extracteur AlethIP. Nous présentons ici les premiers résultats construits sur le corpus LEXTER. Nous avons travaillé sur des séquences N_{Adj} extraites via LEXTER. A partir de 8754 séquences comportant des groupes nominaux, nous avons extraits 586 mots (des noms) auxquels sont attachés 1413 arbres élémentaires de type : S_{n1} -> Nom Adj, S_{n2} -> Adj Nom, S_{n3} -> Adj XX, S_{n4} -> XX Adj. Cette première sélection a donc consisté à ne garder que les arbres binaires portant les feuilles Nom/XX et Adj. Sur ce corpus, le processus de génération conduit à la création des prototypes pour représenter les catégories syntaxiques Nom, Adj, XX, S_{n1}, S_{n2}, S_{n3}, S_{n4}, des objets traits associés et de plus de 2000 prototypes par copie et ajustement. GASPAR a ensuite cherché à repérer les prototypes de mots qui partageaient exactement les mêmes comportements. Ce processus de classement conduit à la création automatique de 54 pôles de comportements partagés. On présente ci-dessous quelques classes de mots obtenues.

Noms partageant un arbre élémentaire (S_n -> Nom Adj):

- (1) occipital bras aisselle epaule Adj ={gauche}
- (2) excès surcharge Adj ={ponderal}
- (3) octobre juillet juin mai mars avril Adj ={dernier}
- (4) besoin tableau Adj ={clinique}
- (5) staff discussion reunion exeresse geste reparation resection revascularisation Adj ={medico-chirurgical}
- (6) equipe solution œdème parenchyme plage cœur tuberculose vascularisation Adj ={chirurgical}
- (7) sommet base Adj ={pulmonaire}
- (8) bloc sillon Adj ={auriculo-ventriculaire}

- (9) expression positivité seringue Adj ={electrique}
- (10) oreillette ventricule retard Adj ={droit, gauche}

Noms partageant un arbre élémentaire (S_n -> Adj Nom):

- (17) majorité variabilité Adj ={grand}
- (28) ballon extension Adj ={petit}
- (33) intention symptôme Adj ={premier}

On note que les noms sont majoritairement plus modifiés à droite qu'à gauche. Les classes produites sont, dans l'ensemble, cohérentes mais ne produisent pas encore des résultats pertinents sur le domaine étudié : certaines classes évidentes ou prévisibles sont mises au jour. La classe de mot associée au pôle n°3 est homogène dans sa relation avec l'adjectif *dernier*, de même pour la classe n°2 dans sa relation avec l'adjectif *ponderal*. La classe n°1, où la relation de localisation qualifie un membre ou une région du corps, est elle aussi cohérente ; pour cette classe, on note que les noms qualifiés ne le sont que pour l'adjectif localisant *gauche* ; à la différence de la classe n°10, celle-ci étant moins homogène. Les classes n°5, 6, 9 regroupent quant à elles des noms sémantiquement plus éloignés. L'examen de la simple relation binaire Nom Adj ne permet pas de décrire complètement les mots. Pour enrichir ce travail de description du comportement des mots puis de leur représentation, il convient évidemment de pouvoir examiner d'autres types de relation binaire puis les relations syntaxiques complexes (lien arbre élémentaire / arbre d'analyse). Il convient aussi d'examiner en détail tous les types possibles de regroupements de mots : certains mots partagent individuellement plus de comportements avec d'autres mots.

7. Approche d'un méta-protocole

GASPAR permet de guider la mise au point de nouveaux résultats : l'utilisateur peut, à chaque étape, examiner les résultats déjà établis pour ensuite soit sélectionner parmi les outils disponibles ceux qui permettent d'améliorer la qualité des résultats, soit agir directement sur les objets construits pour les modifier ou pour les remodeler. La mise en place de regards croisés sur les savoirs représentés dépend directement du méta-regard de l'utilisateur sur les résultats qu'il génère en activant les outils définis : le méta-regard de l'utilisateur et ses interventions guident le travail d'affinement des résultats construits et leur interprétation.

7.1. Un environnement de programmation pour l'expérimentation

L'environnement mis en place avec Self facilite les manipulations concrètes et directes sur les objets. Un utilisateur peut "communiquer" avec les agents du programme directement pour les modifier sans avoir à connaître précisément les mécanismes qui permettent ces modifications. Un programme Self est constitué d'entités "vivantes" et non d'entités inertes et immuables. Pour changer un objet, il est inutile de recompiler, il suffit de modifier l'objet suivant les besoins. Notre approche expérimentale s'accorde bien avec la possibilité de modifier interactivement les représentations créées. L'interface graphique de Self

([Self 95], [Wolczko 1995]) permet une visualisation des objets, et l'utilisateur peut "ausculter" les contenus des différents attributs ou des méthodes associés aux objets. Il peut aussi agir sur ces différentes facettes d'un objet. Nous avons déjà indiqué que certaines phases de travail manuel peuvent être réalisées pour affiner les résultats ou les représentations produits par GASPAR. Pour construire un programme de TALN capable d'agir dynamiquement sur ces mêmes objets, on doit disposer d'outils qui produisent les mêmes résultats mais qui agissent de manière transparente pour permettre une modification dynamique des représentations construites. GASPAR gère automatiquement les savoirs rencontrés et leurs modifications éventuelles. Il peut sélectionner les savoirs à traiter et agir sur eux. Pour réaliser ce type de sélection, GASPAR utilise les méta-outils mis en place par Self pour interroger la structure d'un objet donné [Fleury 1997].

7.2. Réflexion de structure avec Self

Une des caractéristiques de Self est que les objets manipulés sont complètement déterminés par les comportements qui leur sont associés. Cette priorité de représentation permet une extension et une flexibilité dans le développement de nouvelles applications. Self dispose aussi d'une facette fondamentale : on peut souhaiter pouvoir examiner ou manipuler les objets au cours d'un processus de calcul ; on utilise les objets comme des données et non plus seulement comme des entités associées à la représentation d'un concept. Ce type de réflexivité permet de "parler" à propos d'un objet plutôt que de lui parler. Cette action est faite en Self avec des objets appelés miroirs [Cointe 1993]. Ces entités rendent visibles au niveau du langage les informations contenues dans la représentation interne des objets. Un miroir sur un objet x permet essentiellement de faire des requêtes sur la représentation interne de cet objet. Dans la réalité, les miroirs sont des objets qui ne contiennent aucune information en eux-mêmes. Un miroir est simplement un receveur capable de répondre à un ensemble de primitives du système qui donnent accès en lecture aux informations sur l'objet.

7.3. Utilisation des méta-savoirs et contrôle

GASPAR active différents outils qui permettent d'agir à un méta-niveau pour évaluer ou contrôler les objets manipulés. GASPAR utilise les méta-processus capables d'évaluer la structure interne des objets construits pour, par exemple, rechercher et sélectionner parmi tous les objets disponibles un objet particulier avec un type donné d'attribut. Il peut ensuite lui appliquer les opérations voulues. Ces opérations, intégrées aux programmes construits, restent évidemment transparentes pour l'utilisateur. Ce dernier constitue, malgré tout, une "entité" de contrôle indispensable pour évaluer les résultats construits. C'est à lui d'interpréter et d'évaluer toutes les mises à jour réalisées : il peut d'ailleurs lui-même être à l'origine de modifications sur les savoirs représentés. Puisque l'utilisateur joue un rôle central dans notre approche expérimentale, il est important de souligner que si l'on veut qu'il soit capable d'intervenir pour évaluer ou modifier les objets ou les résultats construits, il doit pouvoir suivre pas à pas les différentes opérations

réalisées pour ensuite évaluer les changements d'états des objets manipulés [Habert 1993]. On conçoit donc qu'il soit utile, voire indispensable, de pouvoir suivre ces évolutions potentielles. A cet effet, les processus réflexifs disponibles ont permis de développer des outils de suivi ou de mise au point de l'analyse. Ces outils présentent, de manière raisonnée, les différents états pertinents des traitements réalisés et des résultats construits à chaque étape.

7.4. Limites actuelles des méta-processus

Tous les méta-processus définis n'agissent en fait que de manière structurelle par rapport aux objets. Ils permettent soit d'ajouter un comportement, soit de retirer un comportement, soit enfin de modifier globalement un comportement. Ils ne sont pas encore capables de modifier partiellement la valeur d'un attribut, et notamment des attributs qui portent des procédures. Si l'on doit modifier ce type d'attribut de manière transparente pour l'utilisateur, il convient de modifier globalement la valeur de cet attribut *i.e* réécrire le code associé et le réaffecter à l'attribut visé. Il est clair qu'une telle mise à jour dépasse très souvent les compétences du non spécialiste. Si l'utilisateur peut intervenir de manière interactive sur tous les objets en utilisant l'interface graphique, il reste important de souligner qu'à moins de connaître toutes les finesses de la programmation avec Self, il reste difficile à un utilisateur non spécialiste de ce langage de gérer tout seul ce type de mise à jour. La mise au point de tels outils de mise à jour dynamique et ponctuelle des objets est d'ailleurs loin d'être triviale [Cointe 1993; 1995].

7.5. Mettre en place la génération de nouveaux savoirs

Notre travail vise à permettre aux objets d'apprendre par ajustements successifs de nouvelles connaissances. Dans la phase de génération des prototypes de mots, ces derniers apprennent dynamiquement les informations qu'il est possible de leur associer. Cet apprentissage est réalisé sous la forme d'une relation d'apprentissage d'un maître à un élève. Le maître en l'occurrence se matérialise sous la forme des informations initiales recueillies sur un corpus donné. Le classement produit peut aussi conduire l'utilisateur à intervenir directement sur les résultats pour l'affiner. Cet apprentissage via l'acquisition de connaissances proposées par un maître ne suffit pas [Pitrat 95]. Un programme de TALN doit pouvoir construire de nouvelles connaissances à partir des connaissances connues.

Création de pôles de comportements

La phase de travail qui consiste à repérer les savoirs communs puis à définir et à créer les pôles de comportements partagés peut être assimilée à une première étape dans la construction de nouveaux savoirs. Celle-ci reste à affiner et à développer. Ces regroupements de savoirs partagés par un ensemble de prototypes de mots ou de prototypes d'arbres ne sont en effet pas identifiés ou nommés. L'opération qui consiste à évaluer la ressemblance comportementale entre les mots ne qualifie pas les ressemblances constatées.

Affiner les processus autodescriptifs des objets

Si deux objets (représentant des mots ou des arbres) doivent "dialoguer" pour évaluer leurs descriptions respectives et celles de leurs comportements, il convient que ces objets puissent activer des processus capables de produire ces descriptions. De plus, dans la mesure où les objets sont susceptibles d'évoluer, il convient que les processus d'autodescription des objets puissent être affinés dynamiquement pour tenir compte de ces évolutions potentielles. Self permet la mise en place de mécanismes qui réalisent cette description des attributs d'un objet donné à tout moment de son existence dans le système. La mise en œuvre de telles opérations sur tous les objets de GASPARD reste à faire. De même il convient de poursuivre la mise en place de processus capables d'interroger plus précisément les attributs attachés aux mots et aux arbres associés si l'on veut donner du sens aux savoirs représentés.

Perspectives : des nuages au cristal ?

Nous avons présenté une approche qui vise à établir une représentation dynamique de mots suivie d'une classification progressive de ces mots. La démarche proposée permet d'envisager une classification hiérarchisée, même si celle-ci reste évolutive. Cette démarche s'accommode assez bien avec le modèle de représentation retenu. La PàP encourage une approche de représentation faite de petits sauts successifs qui améliore la qualité de la représentation produite. Cette démarche s'accorde aussi avec la nécessaire approche artisanale que constitue le travail du linguiste dans sa volonté de décrire les comportements des faits de langue [Milner 89]. Concrètement ce travail a permis de produire un premier dispositif qui représente et classe les mots et leurs comportements. Les informations utilisées, à ce stade de notre travail, sont nettement insuffisantes pour produire des résultats significatifs. Le classement opéré prend appui sur des caractéristiques grossières à la fois en raison de contraintes matérielles et de la difficulté à récupérer et organiser les informations à représenter. De nombreux prolongements restent à faire. Sur le plan technique, la couverture des gros corpus doit être réalisée. Ce travail doit nous permettre d'améliorer la qualité du travail de représentation initié par les processus construits. La confrontation de ces résultats avec les connaissances du linguiste ou avec des bases de connaissances extérieures est prometteuse. Les aspects conceptuels n'ont pas été articulés aux phénomènes traités dans les processus construits ; une étude théorique et technique de ces articulations avec les autres niveaux de savoirs représentés doit conduire à étendre et affiner le travail de représentation amorcé. Le traitement de connaissances extra-linguistiques peut aussi compléter et affiner les processus mis en place. Il reste aussi à penser et à mettre en œuvre une interface graphique qui permette à l'utilisateur d'évaluer et d'interpréter les résultats. Ce développement constitue une voie de travail de grande ampleur mais reste en marge des problèmes exposés ici. Self offre d'ailleurs des solutions intéressantes pour réaliser cette interface. Ce travail a aussi permis de tracer des pistes pour une réflexion sur les problèmes complexes liés aux traitements automatiques des faits (évolutifs) de langue. Notre approche des problèmes de classification, d'apprentissage, de réflexivité doit être étendue. La

confrontation de nos choix et de nos résultats avec les travaux établis par ailleurs doit constituer une voie de recherche fructueuse.

Remerciements

Je remercie B. Habert (ELI - ENS de Fontenay-St Cloud) qui a suivi pas à pas les différentes étapes de ce travail. Je remercie également Didier Bourigault (DER-EDF), B. Habert et Adeline Nazarenko pour m'avoir donné accès aux résultats de LEXTER et de ZELLIG. Je n'oublie bien évidemment pas les membres du groupe Self à Stanford et à Sun Microsystems qui ont développé le langage Self et qui ont toujours répondu avec bienveillance à mes sollicitations.

Bibliographie

- [Bentch 1995] Bentch P., Savitch W., (1995), "An occurrence-based model of word categorization", *Annals of Mathematics and Artificial Intelligence* 14, J.C Baltzer AG, Science Publishers, 95.
- [Biber 1993] Biber D., (1993), "Using register-diversified corpora for general language studies", *Computational Linguistics*, volume 19, numéro 2, p. 219-241, 1993.
- [Blaschneck 1994] Blaschek G., (1994), *Object-Oriented Programming with Prototypes*, Springer-Verlag, Berlin, 1994.
- [Bourigault 1994] Bourigault D., (1994), *LEXTER, un logiciel d'Extraction de TERminologie. Application à l'extraction des connaissances à partir de textes*, Thèse, EHESS, Paris
- [Cointe 1993] Cointe P., Malenfant J., Dony C., Mulet P., (1993), *Etude de la réflexion en Self, un langage à prototypes*, rapport LITP.
- [Cointe 1995] Cointe P., Mulet P., Malenfant J., (1995), *Towards a methodology for Explicit Composition of Meta-Objects*, Actes OOPSLA, SIGPLAN Notices.
- [Fleury 1997] Fleury S., (1997), *La programmation à prototypes, un outil pour une linguistique expérimentale. Mise en œuvre de représentations évolutives des connaissances pour le traitement automatique du langage naturel*, Thèse de doctorat, Paris 7-Denis Diderot.
- [Grefenstette 1993] Grefenstette G., (1993), *Automatic Thesaurus Generation from Raw Text using Knowledge-Poor Techniques*, 9th Annual Conference of the university of Waterloo, Centre for the New Oxford English Dictionary and Text Research, Oxford.
- [Grefenstette 1994] Grefenstette G., (1994), *Explorations in Automatic Thesaurus Discovery*, Kluwer Academic Publisher, Dordrecht, The Netherlands, 1994.
- [Habert 1993] Habert B., Fleury S., "Vers des analyseurs réflexifs", TAL, volume 34, numéro 1.
- [Habert 1996a] Habert B., Nazarenko A., (1996), "La syntaxe comme marche-pied de l'acquisition de connaissances: bilan

critique d'une expérience", *Journées d'Acquisition de Connaissances*.

in *Computer Methods and Programs in Biomedicine*, volume 45, p. 117-120.

[Habert 1996b] Habert B., Folch H., (1996), *Les quasi-arbres : un formalisme logique pour exprimer des requêtes en indexation structurée*, Actes ILN'96, Nantes.

[Habert 1997a] Habert B., Bourigault D., (1997), *A Frontier to Root Transducer for the Evaluation of Two Noun Phrase Extractors*, JST-FRANCIL'97, Avignon.

[Habert 1997b] Habert B., Salem A. Nazarenko A. (19), *Les linguistiques de corpus*, Armand Colin, Paris.

[Harris 1970] Harris Z. (1970), *La structure distributionnelle*, in *Langages*, 20, Larousse.

[Harris 1988] Harris Z. (1988), *Language and Information*, Columbia University Press, New York.

[Haton 1991] Haton JP et al., (1991), *Le Raisonnement en Intelligence Artificielle : Modèles, Techniques et Architectures pour les systèmes à base de connaissances*, InterEditions.

[Liebermann 1986] Liebermann H. (1986), *Using Prototypical Objects to implement Shared Behavior in Object Oriented System*, Artificial Intelligence Laboratory, MIT.

[Mikheev 1994] Mikheev A., Moens M. (1994), *Acquiring and Representing Background Knowledge for a natural Language Processing System*, Actes AAAI.

[Milner 1989] Milner J-C., (1989), *Introduction à une science du langage*, Le Seuil.

[Pitrat 1995] Pitrat J., (1995), *De la machine à l'intelligence*, Hermès.

[Resnik 1993] Resnik P.S., (1993), *Selection and Information : a Class-Based Approach to Lexical relationships*, Ph.D Thesis, University of Pennsylvania IRCS Report 1993-42.

[Rosch 1975] Rosch E., (1975), *Cognitive representation of semantic categories*, in *Journal of Experimental Psychology*, volum 104.

[Rosch 1976] Rosch E., (1976), *Classification d'objets du monde réel : origine et représentations dans la cognition*, in *Bulletin de Psychologie*, S. Ehrlich, E. Tulving (Editors), numéro spécial : *La mémoire sémantique*.

[Self 1995] Self Group 1995(1995), *Self 4.0 Read This First*, Sun Microsystems, Inc. and Stanford University.

[Ungar 1987] Ungar D., Smith R.I B., (1987), *SELF : The power of Simplicity*, Actes OOPSLA, SIGPLAN Notices.

[Ungar 1995] Ungar D., (19), *Programing as an experience: The Inspiration for SELF*, Stanford University.

[Wolczko 1995] Wolczko M., Smith R. B., (1995), *Prototype-Based Application Construction Using Self 4.0*, Sun Microsystems Inc.

[Zweigenbaum 1994] Zweigenbaum P., (1994), *MENELAS: an Access System for Medical Records using Natural Language*,