Un analyseur pré-syntaxique pour le levée des ambiguïtés dans des documents écrits en langue naturelle : application à l'indexation automatique

Alain Merle

THESE

présentée à

l' Institut National Polytechnique de Grenoble

pour obtenir le grade de

DOCTEUR INGENIEUR
" Génie Informatique "

par

Alain MERLE

UN ANALYSEUR PRE-SYNTAXIQUE POUR LA LEVEE DES AMBIGUITES DANS DES DOCUMENTS ECRITS EN LANGUE NATURELLE : APPLICATION A L'INDEXATION AUTOMATIQUE.

Thèse soutenue le 22 septembre 1982 devant la Commission d'Examen :

Monsieur	C. BOITET	:	Président
Messieurs	H. ALARDO G. VEILLON	}	Examinateurs
	E. GRANDJEAN	:	Invité

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Année universitaire 1979-1980

Président : M. Philippe TRAYNARD

Vice-Présidents : M. Georges LESPINARD
M. René PAUTHENET

PROFESSEURS DES UNIVERSITES

MM.	ANCEAU François	Informatique fondamentale et appliquée
	BENOIT Jean	Radioéléctricité
	BESSON Jean	Chimie Minérale
	BLIMAN Samuel	Electronique
	BLOCH Daniel	Physique du Solide - Cristallographie
	BOIS Philippe	Mécanique
	BONNETAIN Lucien	Génie Chimique
	BONNIER Etienne	Métallurgie
	BOUVARD Maurice	Génie Mécanique
	BRISSONNEAU Pierre	Physique des Matériaux
	BUYLE-BODIN Maurice	Electronique
	CHARTIER Germain	Electronique
	CHERADAME Hervé	Chimie Physique Macromoléculaires
Mme	CHERUY Arlette	Automatique
MM.	CHIAVERINA Jean	Biologie, Biochimie, Agronomie
	COHEN Joseph	Electronique
	COUMES André	Electronique
	DURAND Francis	Métallurgie
	DURAND Jean-Louis	Physique Nucléaire et Corpusculaire
	FELICI Noël	Electrotechnique
	FOULARD Claude	Automatique
	GUYOT Pierre	Métallurgie Physique
	IVANES Marcel	Electrotechnique
	JOUBERT Jean-Claude	Physique du Solide - Cristallographie
	LACOUME Jean-Louis	Géographie - Traitement du Signal
	LANCIA Roland	Electronique - Automatique
	LESIEUR Marcel	Mécanique
	LESPINARD Georges	Mécanique
	LONGEQUEUE Jean-Pierre	Physique Nucléaire Corpusculaire
	MOREAU René	Mécanique
	MORET Roger	Physique Nucléaire Corpusculaire
	PARIAUD Jean-Charles	Chimie - Physique
	PAUTHENET René	Physique du Solide - Cristallographie
	PERRET René	Automatique

.../...

MM.	PERRET Robert	Electrotechnique
	PIAU Jean-Michel	Mécanique
	PIERRARD Jean-Marie	Mécanique
	POLOUJADOFF Michel	Electrotechnique
	POUPOT Christian	Electronique - Automatique
	RAMEAU Jean-Jacques	Chimie
	ROBERT André	Chimie Appliquée et des matériaux
	ROBERT François	Analyse numérique
	SABONNADIERE Jean-Claude	Electrotechnique
Mme	SAUCIER Gabrielle	Informatique fondamentale et appliquée
M.	SOHM Jean-Claude	Chimie - Physique
Mme	SCHLENKER Claire	Physique du Solide - Cristallographie
MM.	TRAYNARD Philippe	Chimie - Physique
	VEILLON Gérard	Informatique fondamentale et appliquée
	ZADWORNY François	Electronique

CHERCHEURS DU C.N.R.S. (Directeur et Maître de Recherche)

M.	FRUCHART Robert	Directeur de Recherche
MM.	ANSARA Ibrahïm	Maître de Recherche
	BRONOEL Guy	Maître de Recherche
	CARRE René	Maître de Recherche
	DAVID René	Maître de Recherche
	DRIOLE Jean	Maître de Recherche
	KAMARINOS Georges	Maître de Recherche
	KLEITZ Michel	Maître de Recherche
	LANDAU Ioan-Doré	Maître de Recherche
	MERMET Jean	Maître de Recherche
	MUNIER Jacques	Maître de Recherche

Personnalités habilitées à diriger des travaux de recherche (décision du Conseil Scientifique)

E.N.S.E.E.G.

MM.	ALLIBERT Michel	
	BERNARD Claude	
	CAILLET Marcel	
Mme	CHATILLON Catherine	
MM.	COULON Michel	
	HAMMOU Abdelkader	
	JOUD Jean-Charles	
	RAVAINE Denis	
	SAINFORT	C.E.N.G.

.../...

MM. SARRAZIN Pierre
SOUQUET Jean-Louis
TOUZAIN Philippe
URBAIN Georges — Laboratoire des Ultra-Réfractaires ODEILLO

E.N.S.M.E.E.

MM. BISCONDI Michel
BOOS Jean-Yves
GUILHOT Bernard
KOBILANSKI André
LALAUZE René
LANCELOT Françis
LE COZE Jean
LESBATS Pierre
SOUSTELLE Michel
THEVENOT François
THOMAS Gérard
TRAN MINH Canh
DRIVER Julian
RIEU Jean

E.N.S.E.R.G.

MM. BOREL Joseph
CHEHIKIAN Alain
VIKTOROVITCH Pierre

E.N.S.I.E.G.

MM. BORNARD Guy
DESCHIZEAUX Pierre
GLANGEAUD François
JAUSSAUD Pierre
Mme JOURDAIN Geneviève
MM. LEJEUNE Gérard
PERARD Jacques

E.N.S.H.G.

M. DELHAYE Jean-Marc

E.N.S.I.M.A.G.

MM. COURTIN Jacques
LATOMBE Jean-Claude
LUCAS Michel
VERDILLON André

REMERCIEMENTS

Je tiens à remercier :

- H.ALARDO, de la société TELESYSTEMES, pour avoir accepté de juger ce travail.

- C.BOITET, maître de conférence à l'Université de Grenoble, pour m'avoir fait le grand honneur de présider le jury de cette thèse.

- E.GRANDJEAN, ingénieur C.N.R.S., pour sa gentillesse et sa disponibilité; ce travail est le fruit d'une étroite collaboration avec lui.

- G.VEILLON, professeur ENSIMAG, pour toute l'aide qu'il m'a apporté lors de la réalisation de cette thèse et pour l'intérêt qu'il a manifesté à mon travail.

Je tiens aussi à remercier toutes les personnes que j'ai cotoyées pendant ces trois dernières années et qui ont toujours répondu avec une extrême gentillesse à toutes mes questions. En particulier, j'adresse tous mes remerciements à P.BERLIOUX, Y.DESCOTTE et M.LEVY pour l'aide qu'il m'ont apportée lors de l'écriture de cette thèse, et à D.DUJARDIN pour son aide lors de la réalisation du logiciel PIAFPS.

Je tiens enfin à remercier D.IGLESIAS et toute son équipe pour le soin apporté à l'impression de cette thèse.

SOMMAIRE

SOMMAIRE

INTRODUCTION

Le traitement automatique des langues a fait l'objet de développements importants ces dernières années.

Initialement orienté vers des applications déjà ambitieuses telles que la traduction automatique (VAUQ), les progrès de l'intelligence artificielle ont fait apparaître un intérêt croissant pour des applications orientées vers la "compréhension" de textes en langue naturelle.

Parallèlement, les études menées sur le traitement de textes importants ont été poursuivies.

Dans ce domaine, nécessitant une analyse plus formelle de la langue, les systèmes s'organisent généralement autour d'analyseurs dirigés par la syntaxe, suivant des techniques analogues à celles de la compilation.

Les difficultés combinatoires (ambiguïtés) et les problèmes sémantiques ont conduit à proposer des modèles équivalents fondés sur des techniques procédurales (WIN,WOl). Malgré les différents procédés, on constate, cependant, une convergence pour l'obtention d'une représentation de dépendances conceptuelles des phrases analysées, équivalente aux structures de dépendances. Cette représentation permet notamment une interprétation fonctionnelle de la structure de la phrase, facilement traitable par des langages informatiques.

Cette approche syntaxique, pose quelques problèmes. La modélisation de la grammaire de la langue par une grammaire "hors-contexte", bien qu'elle apparaisse comme indispensable, crée une complexité parfois incompatible avec les contraintes de certaines applications (Si n est le nombre de mots de la phrase, le traitement est une fonction de n^3). La présence de nombreuses homographies, réelles ou engendrées par l'insuffisance des modèles de description, accroit le caractère combinatoire de cette analyse.

Ces remarques ont donné naissance à des systèmes non-syntaxiques de résolution des homographies. Mais, à notre connaissance, ils sont basés sur des traitements statistiques (FLU). A cause du risque d'erreur inhérent à ces méthodes, ils sont inadaptés à certaines applications.

Il existe tout une classe de problèmes qui demandent une détermination de "parties du discours" dans des textes libres. Outre la traduction automatique, il faut citer la documentation automatique (et plus généralement la bureautique), la génération de la parole, etc...

Actuellement l'impossibilité d'effectuer une analyse syntaxique complète et fiable et d'obtenir une "compréhension" suffisante de textes libres (en vue, par exemple, d'une interprétation des textes par un logiciel), provoque une situation de blocage.

Il semble pourtant que la majorité des textes présente des constructions simples, ne nécessitant qu'un modèle d'états finis. De même, si certaines ambiguïtés sont effectivement complexes et demandent une analyse complète de la phrase, la plupart apparaît évidente à un lecteur humain.

Exemple

Dans la phrase :

"La maison de l'oncle que nous avons vu".

Seule une analyse syntaxique complète de la phrase permet de savoir ce que qualifie "vu" : la maison ou l'oncle.

Par contre les homographies générées par les mots "marche" (substantif ou verbe) et "élèves" (substantif ou verbe) apparaissent évidentes à un lecteur humain dans les phrases :

"je marche ..."
"Le professeur demande aux élèves de réciter leur leçon".

(nous ne nous intéressons ici qu'aux problèmes syntaxiques : "demande" et "élèves" sont-ils des verbes ou des substantifs ?)

Ces problèmes semblent pouvoir se résoudre très simplement, sans faire appel à une analyse suivant le modèle "hors-contexte", ni pour la résolution des homographies, ni, semble-t-il, pour la construction de cette dernière phrase.

Ces remarques ont amené à la définition d'un prétraitement à l'analyse syntaxique de textes libres. Ce traitement, par définition simple et rapide, doit permettre de résoudre tout ce qui ne demande pas le classique modèle "hors-contexte", notamment les ambiguïtés évidentes de la langue.

Nous nous plaçons, dans cette étude, dans un environnement d'analyse formelle de textes libres ("full text") ne demandant qu'une compréhension limitée. Le but du système présenté dans cette thèse est de résoudre certains des problèmes liés à l'analyse syntaxique des

langues naturelles. Dans l'échelle classique des traitements formels nous nous plaçons entre une analyse morphologique (reconnaissace des mots de la langue) et une analyse syntaxique (reconnaissance des constructions). Nous avons donc adopté une approche pragmatique de ces problèmes dont le but principal est l'efficacité.

Ces travaux ont été motivés par une application d'indexation automatique de documents destinés à une base documentaire. Ils ont été menés en collaboration avec la société TELESYSTEMES.

Dans le premier chapitre nous effectuons un rapide tour d'horizon des différentes méthodes d'analyse formelle des langues naturelles et nous présentons les problèmes que pose leur emploi, et leurs limitations. Nous présentons également quelques remarques qui furent le point de départ de cette réalisation.

Les chapitres 2 et 3 décrivent le système PIAFPS. Le chapitre 2 est une présentation des techniques et des mécanismes que nous avons mis en oeuvre ainsi que de leur puissance théorique. Le chapitre 3 décrit l'implantation de ce système dans le cadre d'une application d'indexation automatique de documents destinés à une base documentaire.

Le chapitre 4 est une discussion à partir des expérimentations que nous avons menées. Il présente également les évolutions possibles du système PIAFPS.

CHAPITRE 1 : SITUATION

Ce chapitre n'a pas pour but de faire une analyse précise et exhaustive des systèmes d'analyse des langues naturelles. Nous y présentons rapidement certaines méthodes mises en oeuvre dans ces systèmes et précisons les raisons pour lesquelles il nous était impossible de les utiliser dans le cadre de l'application d'indexation automatique décrite au chapitre II.

1) GENERALITES

Formellement, la langue écrite est un ensemble de chaînes de caractères. Toute étude des langues naturelles passe d'abord par une segmentation de ces chaînes de caractères en "mots". L'étude proprement dite (ainsi que généralement la "grammaire" de la langue) concerne le rôle de ces mots dans un texte. Ce premier traitement est appelé la morphologie.

Outre cette segmentation, la morphologie délivre pour chacun de ces mots un certain nombre d'attributs. Ces attributs véhiculent les informations repérables à partir de la chaîne de caractères et seront utilisés lors des traitements ultérieurs. En tant que tels ce ne sont pas des constantes de la langue : ils sont définis en fonction de plusieurs paramètres parmi lesquels on peut citer :

- Le modèle morphologique pour le choix de la représentation de ces informations.

- Le modèle de l'analyse suivante. C'est lui qui utilise ces informations et donc qui décide de leur nature.

D'une façon générale nous pouvons représenter un mot comme un ensemble d'attributs.

Exemple

Si nous voulons réaliser une analyse syntaxique du texte, les attributs le plus souvent utilisés sont :

- "La catégorie syntaxique" du mot : Verbe, Substantif, Adjectif, ...

- Certaines variables : Genre (masculin, féminin, neutre), Nombre (singulier, pluriel), ...

- etc

Pour une analyse sémantique du texte, on retiendra en priorité certains traits sémantiques, par exemple :

- Verbe d'action, d'état.
- Objet, substance.
- Animal, être humain.
- etc...

Il existe de nombreuses méthodes d'analyse de langues que nous pouvons regrouper en trois grands niveaux :

- L'analyse morphologique.
- L'analyse syntaxique.
- L'analyse sémantique.

Sans entrer dans les détails, ni donner de définitions très précises, la sémantique s'intéresse au sens des textes analysés et essaie d'extraire les concepts véhiculés par la langue. La syntaxe s'intéresse plus au contenant qu'au contenu, et analyse la construction grammaticale des textes.

Ces trois modèles sont soit concurrents, soit complémentaires : Les systèmes question-réponse se désintéressent de la syntaxe au profit de la sémantique alors que certaines analyses formelles (le système PIAF (COU), (JOL), (LOP) entre autres) réalisent ces trois types de traitements.

L'analyse morphologique est indispensable et la plupart des systèmes y ont recours. Les modèles ultérieurs sont destinés à fournir une certaine "compréhension" du texte, pour cela ils doivent travailler sur un ensemble fini et autant que possible réduit de symboles d'entrée. La morphologie doit donc regrouper les mots en classes utilisables pour ces traitements, la définition des classes variant en fonction des traitements envisagés.

Si l'on veut décrire l'agencement des mots les uns par rapport aux autres de manière finie, il est nécéssaire de créer des classes d'équivalence comprenant les mots qui jouent un rôle identique dans cet agencement. A chaque classe est associée une étiquette que l'on appellera catégorie grammaticale ou catégorie lexicale. Les règles d'agencement portent donc sur ces étiquettes qui sont en nombre restreint et non plus sur les mots eux-mêmes. Si une même chaîne de caractères se présente en deux occurrences dans un contexte de valeurs grammaticales différentes, on peut lui attribuer des étiquettes grammaticales différentes. Dans le cas où un mot appartient à plusieurs classes grammaticales, on dit qu'il y a une ambiguïté grammaticale ou une homographie. Bien sur, ces homographies sont inhérentes au modèle car la classification et surtout le choix des classes sont arbitraires.

Exemple

"La suspension de ce véhicule entre en résonnance entre 80 et 90 km/h".

Le premier "entre" est un verbe alors que le deuxième est une préposition. La position des valeurs grammaticales des autres mots de la phrase permet seule de donner l'étiquette syntaxique du premier et du deuxième "entre". La classification en "verbe" ou "préposition" pour le mot "entre" est totalement arbitraire, elle est due au choix du modèle morphologique. A priori, rien n'empêche de définir une classe pour le mot "entre", nous éliminerions ainsi l'homographie mais multiplierions les classes les rendant sans doute inutilisables aux analyses ultérieures (dans ce cas précis à une analyse syntaxique).

Ce type d'ambiguïté est très courant en français et provient soit d'une coincidence comme c'est le cas pour : entre, car, or, etc..., soit d'une dérivation sémantique plus ou moins lointaine comme c'est le cas entre beaucoup de verbes du premier groupe et des substantifs : programme, marche, hache, etc.... Ces ambiguïtés grammaticales ne sont pas propres au français, on en rencontre aussi dans la plupart des langues, surtout quand elles ont un système de dérivation très réduit.

Exemple

En anglais : "John can open his can of beans"

où "can" est respectivement verbe et substantif.

Il faut noter que même les langues avec un fort degré de déclinaison, tel le russe, présentent ce genre d'ambiguïtés (le système flexionnel en génère lui-même).

Le phénomène que nous venons de décrire pour la syntaxe se produit aussi au niveau sémantique. Selon le contexte, un mot peut être interprété de plusieurs manières.

Exemple

"Le facteur dépose les lettres."
"Le facteur est multiplié par 3."

Dans ces deux exemples, "facteur" correspond à deux concepts différents, même s'il y a une origine étymologique commune.

Ces deux types d'ambiguïté ne se recouvrent pas. Souvent une ambiguïté syntaxique implique une ambiguïté sémantique (Exemple : hache, porte, règle, fin, or, car, etc...) dans d'autres cas (Exemple : programme, anesthésie, éponge, etc...) le rôle syntaxique des mots est différent mais une partie importante du sens est conservée.

En fait, une langue naturelle, contrairement aux langages formels, n'est pas définie par sa grammaire. La nécessité de passer par une analyse non globale de la phrase conduit à attribuer plusieurs significations aux mots, ces significations étant non ambigues dans le contexte. La grammaire, ou tout autre modèle d'analyse de la langue, n'est qu'une approximation de cette langue, et par là même est imparfaite.

Le nombre d'ambiguïtés est lié à la finesse de définition des mots aussi bien pour la syntaxe que pour la sémantique. Il est alors parfois difficile de déterminer si une ambiguïté est une ambiguïté de la langue ou si elle est générée par le modèle de description de la langue. Le choix des classes est, nous l'avons dit, totalement arbitraire; il oscille entre les trois extrêmes :

- Pratiquement une classe par mot, ce qui a l'avantage de supprimer à peu près les homographies mais devient totalement inutilisable pour la suite; par exemple une classe pour chacune des chaînes de caractères généralement homographes suivantes : entre, marche, hache, ... Il faut noter que l'inverse de cette solution est également possible, une seule classe d'équivalence pour tous les mots mais cette "solution" est sans intérêt si ce n'est pour une analyse uniquement morphologique destinée à répondre à la question : cette chaîne de caractères est-elle un mot du langage ?

- Une classification très précise; mais alors nous multiplions plus ou moins artificiellement les homographies. Par exemple, nous pouvons définir les classes suivantes : substantif masculin singulier, substantif masculin pluriel, substantif féminin singulier, substantif féminin pluriel, ...

  Exemple

  Les adjectifs "rouge", "pâle",... seraient ambigus entre les deux catégories "adjectif qualificatif masculin" et "adjectif qualificatif féminin". Cette homographie apparaît comme artificielle car le sens de ces mots ne change pas dans les deux classifications.

- Une classification plus large, qui correspond à un regroupement des classes précédentes; par exemple regroupement de tous les substantifs en une classe. Cette solution a le mérite de réduire le nombre d'homographies, mais au risque de perdre de l'information.

  Exemple

  Nous avons ainsi éliminé les homographies sur "rouge" et "pâle", mais, a priori, nous avons perdu l'information du genre de tous les adjectifs et de tous les substantifs; "vert" sera un adjectif et non plus un "adjectif masculin".

L'analyse syntaxique a pour rôle d'associer une structure à la phrase (ou toutes les structures possibles). Elle permet de lever la plupart des ambiguïtés et, si c'est nécessaire, elle contrôle la validité d'un texte.

Certains auteurs nient l'utilité d'une syntaxe formelle au profit d'une analyse sémantique; il apparaît pourtant qu'elle est sans doute indispensable à une analyse de textes libres. Les systèmes question-réponse, ou plus généralement les systèmes basés sur un dialogue, peuvent s'en passer : il n'est pas nécessaire de connaître la structure des phrases pour en saisir le sens car le contexte guide l'analyse. Mais les analyses formelles de textes libres indépendantes de tout contexte de production et d'interprétation ne peuvent probablement pas se passer d'une telle analyse syntaxique, en raison, d'une part des informations structurelles qu'elle permet de fournir, et d'autre part en raison des ambiguïtés qu'elle est la seule à pouvoir lever.

Exemple

Seule la syntaxe permet de résoudre l'ambiguïté sur le mot "vu" (qualifie-t-il la maison ou l'oncle ?) dans la phrase :

"La maison de l'oncle que nous avons vu"

L'analyse sémantique peut être considérée comme l'étape suivante dans la compréhension d'un texte. Sans entrer dans les détails, ni donner de définition précise, nous pouvons dire qu'elle permet d'extraire les concepts d'une partie d'un texte et d'en saisir ou formaliser le sens. La difficulté de modélisation d'une langue naturelle entraine de grandes difficultés dans le traitement général de textes libres. Certaines réalisations, dans le cadre d'une langue réduite à un domaine particulier, présentent un grand intérêt; ce sont, entre autres, le système SAM et le modèle conceptuel de SHANK ((SH1), (SH2)), le système de WILKS ((WI1), (WI2)) (bien que ce dernier ne soit pas un réel système de compréhension), le système BARRY (COLB). Dans un domaine plus général, il faut citer les travaux de POGNAN (POG) orientés vers la compréhension implicite de textes scientifiques. JOLOBOFF (JOL) et LOPEZ (LOP) ont réalisé à partir du système PIAF une analyse sémantique formelle de textes libres, mais elle aussi a montré les difficultés de manipulation des concepts sémantiques et le niveau de compréhension obtenu était insuffisant pour un traitement général. Ces analyses sont d'autre part extrèmement complexes, ce qui nuit à leur utilisation dans le cadre de certaines applications.

2) ANALYSE SYNTAXIQUE

Nous avons dit que l'analyse syntaxique d'un texte était probablement indispensable pour l'analyse de textes libres, mais cette analyse pose de nombreux problèmes liés à la modélisation de la langue. Il faut en effet remarquer que , contrairement aux langages de programmation, une langue n'est pas définie par sa syntaxe; la syntaxe est écrite a posteriori et en est une approximation. Dans ce cadre il parait difficile de faire une analyse précise et cependant assez simple.

En compilation, les langages de programmation sont, en général décrits à l'aide d'une grammaire "hors contexte". Les algorithmes d'analyse sont alors élaborés en fonction de cette grammaire. Deux grandes classes d'analyseurs ont alors vu le jour. Ceux qui sont apparus autour des années 1965, que l'on pourrait appeler analyseurs généraux et qui s'appliquent sur des grammaires "hors-contextes" presque quelconques (élimination du symbole vide, de symboles inaccessibles, etc...). Leur temps d'exécution étant en général de l'ordre de n3, où n est le nombre d'éléments de la phrase. On a défini depuis des algorithmes plus performants qui ne s'appliquent que sur une classe particulière de grammaire "hors contexte" (Bounded context, LL(K), précédences) ((FLOYD), (KNU3), (KNU4)). La philosophie est la suivante : à partir d'une grammaire vérifiant certaines conditions, on établit des tables, ou des matrices, qui permettront ensuite à l'exécution d'obtenir des analyseurs déterministes. Les résultats obtenus sont spectaculaires et la plupart des compilateurs actuels utilisent maintenant une des méthodes particulières évoquées précédemment. Est-il possible d'utiliser ces analyseurs (généraux ou particuliers) dans le cas des langues naturelles ?

2.1) Les analyseurs généraux et particuliers

Plusieurs essais ont été faits d'après la définition de classes de grammaires pour les langues naturelles proposée par CHOMSKY ((CHO)). L'inconvénient des analyseurs généraux, outre leur complexité, réside dans le nombre de structures proposées. Ce nombre est directement lié aux difficultés de modélisation de la grammaire d'une langue naturelle. Malgré certaines améliorations destinées à pallier ces inconvénients (grammaire à saturation par exemple), la maîtrise de ce type d'analyse reste assez délicate et se pose le problème de la convergence de la grammaire que l'on modifie en fonction des textes à analyser.

Les analyseurs particuliers ne s'appliquent que sur une certaine classe de grammaires "hors contextes". Le principe est en général le

suivant : en fonction d'une grammaire hors contexte G, il faut trouver une grammaire hors contexte équivalente G' qui vérifie certaines conditions. Comme il n'existe pas d'algorithme qui permette de trouver cette grammaire automatiquement (le problème est indécidable), on opère, en pratique, par transformations successives en examinant, à la main, les règles qui font que la grammaire ne satisfait pas aux conditions. Ce travail peut être assez long, voire dans quelques cas impossible, mais lorsqu'on a trouvé G', l'algorithme proposé est alors très performant. Dans les cas de langages de programmation, cette approche est très intéressante et est utilisée par la majorité des compilateurs actuels du fait que le langage que l'on désire compiler est bien défini et qu'il suffit de trouver une grammaire équivalente satisfaisant aux critères. Par contre, dans le cas des langues naturelles, le langage n'est pas défini par sa grammaire. Les règles de grammaire ne sont qu'approximatives et on est amené à y apporter sans cesse des modifications à chaque analyse d'un nouveau texte. Comme il n'existe pas d'algorithme de transformation de grammaire, il est impensable, dans ce cas, d'utiliser une méthode particulière.

2.2) Grammaires

Dans le cas où l'on admet que le texte d'entrée est correct, l'écriture des règles ne pose pas de problème. Par contre, si l'on admet que le texte peut être incorrect et si l'on désire détecter le maximum d'erreurs, il faut établir ces règles avec soin. En effet l'emploi de règles récursives ne permet pas toujours la détection des formes erronées.

En outre, les structures obtenues à partir des grammaires hors contextes sont difficilement interprétables : c'est-à-dire inutilisables par d'autres modèles. Il faut donc utiliser un algorithme de transformation de structures qui permet l'obtention de structures plus facilement utilisables (interprétables), telles que les structures de dépendances.

Les grammaires génératives (voire transformationnelles), sont des modèles parfois satisfaisants du point de vue de la théorie linguistique (explication du phénomène). Ces modèles sont en général très satisfaisants pour la description et l'analyse des langages artificiels. Par contre dans le cas de l'analyse automatique des langues naturelles, ils posent certains problèmes.

Il apparaît, à la suite des remarques précédentes, que l'approche des grammaires génératives pour l'analyse des langues naturelles est difficilement réalisable. Aussi certaines réalisations originales ont vu le jour. Parmi celles-ci il faut citer l'analyse de dépendances du système PIAF ((COU), (VEILLI)) et les travaux effectués à partir des ATN (Augmented Transition Networks) ((WO1), (WO2)). Cette dernière

approche correspond à une présentation plus algorithmique des problèmes, la modélisation de la grammaire pouvant être représentée par un réseau d'automates.

2.3) Analyse de dépendances

Dans le système PIAF, l'analyseur n'essaie pas de construire une ou plusieurs structures représentatives du texte, mais, par contre, essaie d'éliminer toutes les structures impossibles. Ce mécanisme de filtrage permet d'exprimer simplement les relations de dépendances entre les mots, et d'éliminer une grande partie des structures "parasites" que produirait une grammaire générative. Ces grammaires de dépendances sont équivalentes à des grammaires hors-contexte.

Dans PIAF, la grammaire est représentée sous forme de relations de dépendances qui décrivent toutes les dépendances possibles entre un gouverneur et ses dépendants. Chacun de ces objets est représenté par sa catégorie syntaxique. Les dépendances sont notées sous la forme de valeurs numériques arbitraires. Ces valeurs numériques permettent d'imposer certains positionnements des différents dépendants d'un même gouverneur. Les structures de dépendances sont alors facilement interprétables.

Exemple

Considérons l'éternel exemple de la phrase "Le petit chien noir mange la soupe", et les relations de dépendances :

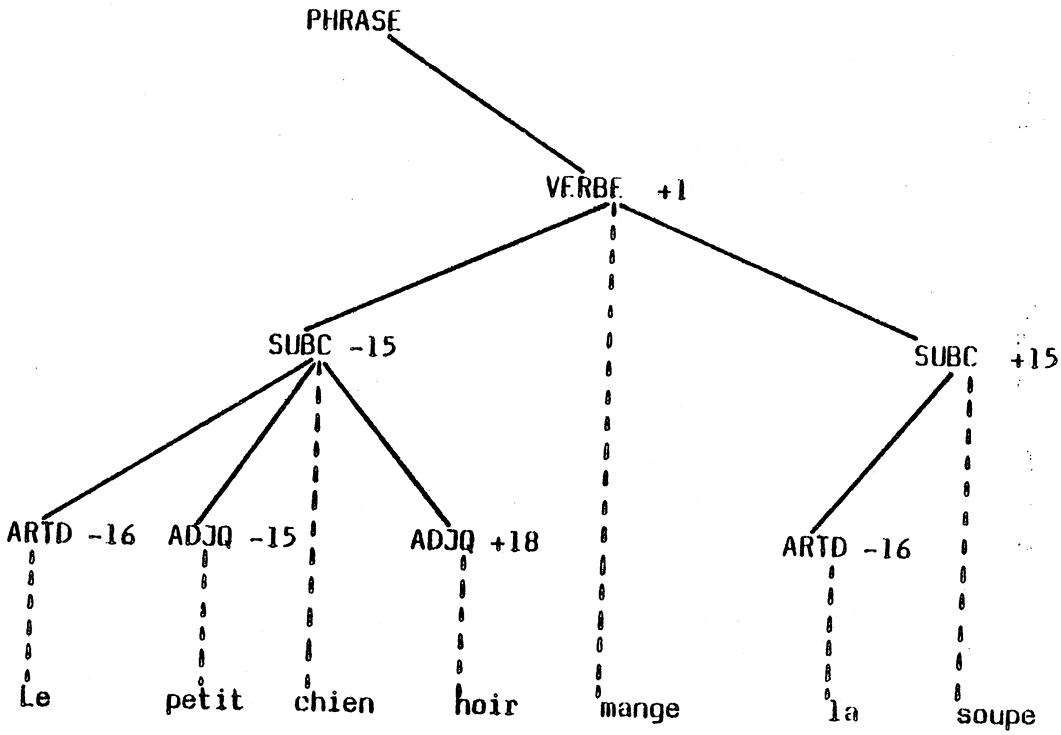
(1) PHRA * VERB := +1 ; Cette relation indique que le gouverneur principal est un verbe.

(2) VERB * SUBC := -15,+15; Cette relation précise qu'il peut y avoir à gauche (-15) ou à droite (+15) d'un verbe, un substantif.

(3) SUBC * ARTD := -16; Cette relation précise qu'à gauche d'un substantif, on peut trouver un article.

(4) SUBC * ADJQ := -15,+18; Cette relation précise qu'à gauche et à droite d'un substantif, on peut trouver un adjectif qualificatif.

La structure obtenue, avec les poids correspondants est :



Cette approche "mécanisme de filtrage", a permis de compléter ces traitements par un module de contrôle d'accord des variables, fonctionnant sur le même principe (élimination de structures impossibles) afin de réduire le nombre de structures proposées. PIAF ne limite ni le nombre ni la nature des filtres.

3) CONCLUSION

L'analyse sémantique de textes libres nous paraît, actuellement, irréaliste en raison des difficultés de modélisation et de manipulation des concepts sémantiques, de sa complexité, et donc d'un niveau de "compréhension" très insuffisant pour un traitement général. Ce type d'analyse donne cependant de bons résultats dans certains domaines particuliers, c'est-à-dire lorsque la langue est simplifiée et les concepts manipulés réduits à une application donnée, et lorsque le contexte de production peut guider ces analyses.

L'analyse syntaxique, en permettant de construire la structure

d'une phrase, semble être, pour l'instant, le traitement le plus élaboré utilisable pour des textes libres. Cependant cette analyse, mise en oeuvre sur les résultats d'une analyse morphologique génératrice d'ambiguïtés, ne peut, dans bien des cas, être employée à cause de sa complexité et de son caractère fortement combinatoire. Cette complexité est liée d'une part à la modélisation de la langue (modèle hors-contexte), d'autre part à la combinatoire engendrée par les ambiguïtés. Il nous semble assez irréaliste d'utiliser un tel analyseur pour résoudre les homographies. Il faut cependant remarquer que certaines homographies ne peuvent être résolues que grâce à une analyse complète de la phrase.

Exemple

Soit la phrase : " La petite marche dans la cour". Si nous supposons que nous n'acceptons que les phrases comportant un verbe, l'ambiguïté sur "marche" (substantif ou verbe) ne peut être levée que par l'analyse complète de cette phrase, les deux catégories lexicales correspondant à des structures localement valides.

Par contre dans la phrase : "Je marche dans la cour", "marche" est toujours ambigu mais la présence de "je" permet de décider immédiatement le choix de la CL "verbe", sans analyse de la phrase.

Pour être utilisée dans des contextes demandant des traitements rapides, cette analyse syntaxique doit être accélérée par rapport aux réalisations actuelles. En effet, et nous le verrons en détail dans le paragraphe 4, nous pensons que la plupart des problèmes peuvent se résoudre localement sans une analyse globale de la phrase.

4) REMARQUES

4.1) Remarques générales

La lenteur des analyseurs actuels est due d'une part à la modélisation de la grammaire du langage, et, d'autre part, aux ambiguïtés. La modélisation est à notre avis imparfaite, en effet, une langue naturelle telle que le français ou l'anglais est en majorité linéaire (c'est-à-dire analysable avec un modèle d'états finis). En français, les seules constructions récursives, et donc pouvant demander le modèle hors-contexte, sont :

- Les relatives.
- Les compléments de nom.
- Les incises.
- La coordination.

De toutes façons le niveau d'imbrication, au moins pour la langue parlée, est limité à trois ou quatre parenthèses par la compréhension du lecteur.

Ces remarques amènent à penser qu'il doit être possible de construire partiellement la structure d'une phrase et par là même de résoudre certaines ambiguïtés avec un modèle moins coûteux que le classique modèle hors contexte, c'est-à-dire avec un modèle d'états finis. Cette analyse, par définition, partielle et, dans la plupart des cas, incapable de construire une structure complète de la phrase, pourrait définir la structure de certains groupes syntaxiques (ou portion de phrase) et résoudre certaines homographies dont le voisinage est simple. Cette idée est assez classique et a déjà été employée dans certaines réalisations ((VAUQ), (VEILL1)).

La seconde cause de la lenteur des analyseurs est la présence des homographies qui accroissent le caractère combinatoire de l'analyse. Il est sans doute possible d'en résoudre la plupart avec un traitement local, de surface et très rapide.

Nous supposons, et c'est le cas de tous les systèmes résolvant les homographies, que le texte analysé est correct. Les seules incohérences sont dues à une mauvaise interprétation ou, plus exactement, à un mauvais choix lors de la résolution des homographies. Par exemple pour la phrase "je marche", qui est intrinsèquement correcte, le choix de la catégorie "substantif" pour "marche" crée une incohérence. Notre but sera de déceler ces incohérences.

La référence étant le texte, nous prendrons la démarche inverse des grammaires génératives, c'est à dire, plutôt que d'essayer de répondre à la question "Telle chaîne appartient-elle au langage", nous avons décidé d'essayer de répondre à la question "Telle chaîne appartient-elle au complémentaire du langage". Ces deux propositions semblent équivalentes, mais la différence essentielle réside dans leur mise en oeuvre: rechercher si une chaîne appartient au langage demande une analyse de toute la chaîne; rechercher une non appartenance peut être effectué avec un traitement local : l'erreur ou l'incohérence est due à une homographie "mal" résolue, nous pouvons donc essayer de déceler les incohérences dans le voisinage de cette ambiguïté. Une telle méthode nous autorise, de plus, à employer un modèle plus large que la grammaire du langage, c'est-à-dire à reconnaître un langage plus restreint que le complémentaire du langage initial. Il est probable qu'une telle méthode ne définit qu'un sur-ensemble du langage, mais notre objectif est de définir un sur-ensemble non-ambigu. Comme nous n'examinons que les homographies, les réponses à la question initiale seront :

"la chaîne appartient au langage reconnu, c'est-à-dire à un sous-ensemble du complémentaire de la langue", donc le choix de telle CL pour telle homographie est mauvais.

"la chaîne n'appartient pas au langage reconnu", le choix est possible, mais nous ne pouvons pas affirmer que la chaîne appartient à la langue.

Exemple

Si dans une phrase nous rencontrons la séquence "je marche", le choix de la catégorie "substantif" pour "marche" est une erreur. Nous pouvons déceler facilement cette incohérence et donc reconnaître que la chaîne "je marche (SUBC)" n'appartient pas à la langue.

Si par contre nous rencontrons la chaîne "la petite marche dans la cour", et que nous n'acceptons que les phrases verbales, les deux choix pour "marche" du substantif ou du verbe sont localement valides. Nous ne décelons donc pas l'erreur qu'entraine le choix de la catégorie "substantif".

Ce type de raisonnement "par l'absurde" est la démarche adoptée dans la démonstration automatique de théorèmes (NILSI), où, pour prouver qu'une propriété est vraie, on cherche à déceler les incohérences de son opposée.

Dans le même ordre d'idée, il faut citer les réalisations fondées sur des traitements statistiques ((FLU)). Le but de ces analyseurs n'est plus de "comprendre" la phrase, mais de résoudre les ambiguïtés repérées par la morphologie. Pour cela, à partir d'une étude statistique des successions de catégories syntaxiques dans un domaine d'application, le système choisit, pour chaque ambiguïté rencontrée, la catégorie ayant la plus forte probabilité d'apparition. Cette méthode consiste donc uniquement en une analyse de surface sans interprétation, ce qui présente de nombreux avantages pour certaines applications, avantages liés principalement au temps d'exécution et à la facilité de mise en oeuvre (programmes à apprentissage). Elle présente aussi certains inconvénients inhérents à l'analyse statistique, le résultat fourni par le système ne peut être assuré dans tous les cas.

4.2) Fondements linguistiques de la méthode

Dans le but de résoudre simplement les homographies, certaines remarques sur le français lui-même nous incitent à penser qu'une analyse de surface peut donner de bons résultats.

Il faut d'abord remarquer que pour un lecteur humain la plupart des homographies n'existent pas : d'une part le contexte dirige la compréhension, d'autre part le lecteur ne s'intéresse pas, ou très peu,

au niveau syntaxique des textes lus.

Nous l'avons dit au paragraphe 4.1, la plupart des langues naturelles sont fortement linéaires c'est-à-dire que, dans la plupart des cas, les groupes syntaxiques sont connexes et les incises très rares. Ceci devrait permettre de déterminer ces groupes localement si nous avons un moyen d'en localiser le début ou la fin.

Il existe en français un grand nombre de marqueurs syntaxiques; ce sont des mots n'apportant aucune information sémantique mais destinés à relier et introduire les mots "importants" et à assurer une certaine construction des phrases. Ce sont en français l'article, la plupart des prépositions, les pronoms personnels, les pronoms relatifs, etc... Certains de ces termes ont des propriétés remarquables par les constructions qu'ils impliquent :

Exemple:

"dans" : introduit un groupe nominal.
"de" : introduit un groupe nominal ou infinitif.
"du" : introduit un groupe nominal.
"je" : introduit un groupe verbal.
"de le" : introduit un groupe infinitif.
"de la" : introduit un groupe nominal ou infinitif.

La présence de ces termes nous permet de prédire le type du groupe syntaxique qui les suit dans le texte. Une fois ce type trouvé, certaines catégories syntaxiques se trouvent interdites car elles ne peuvent appartenir à ce groupe.

Exemple

- Un verbe ne peut suivre "du", car "du" introduit un groupe nominal et un verbe ne peut appartenir à un tel groupe. Par exemple dans la phrase "Il parle du président de ...", "président" ne peut être le pluriel du verbe "présider".

- Un substantif ne peut suivre un pronom personnel sujet (je, tu, il, ...) car ce terme introduit un groupe verbal et un substantif ne peut apparaître dans un groupe verbal. Par exemple dans la phrase "Je marche dans la rue", "marche" ne peut être un substantif.

- etc...

Ces points sont très positifs pour notre application; en effet l'importance de l'ordre des mots dans le texte et son aspect fortement linéaire devraient valider un traitement local d'états finis. La présence, surtout en français, des mots introductifs de groupes syntaxiques devrait nous permettre de situer facilement le début de ces groupes et donc, si nécessaire, d'essayer de lancer un traitement local

à partir d'une limite sûre. En effet, la principale difficulté des traitements locaux est un problème de positionnement dans le texte qui est une structure continue. Pour lancer une analyse partielle, il faut être capable de repérer, sans analyse, les éléments à partir desquels l'analyse va pouvoir se faire, c'est à dire des bornes sûres correspondant aux états initiaux des automates. Dans le cas d'une analyse syntaxique locale et partielle, il faut pouvoir repérer les débuts des groupes syntaxiques sans analyser la phrase complète ni le texte se plaçant avant ces groupes.

Il faut remarquer qu'un grand nombre d'homographies se résout immédiatement en fonction de leurs contextes immédiats (deux ou trois mots). Par exemple dans les phrases comportant les séquences "je marche", "aux élèves", "je la hache",... le choix de certaines catégories est éliminé immédiatement et, bien souvent, il n'en reste qu'une possible. Par exemple dans la phrase "je marche", une fois que nous avons écarté pour "marche" la catégorie "substantif", seule "verbe" est possible. Le texte étant correct, il existe une solution. La catégorie à choisir est donc "verbe".

Certains mots en français ont un comportement très particulier, tels les verbes transitifs et intransitifs, les adjectifs se plaçant avant, après ou indifféremment avant ou après le substantif, les adjectifs substantivables ou non, etc.... Ces informations pourraient être utiles dans un système pouvant les utiliser simplement. De même, nous pouvons remarquer que certaines catégories syntaxiques ne peuvent se suivre en français. Nous pouvons citer par exemple un pronom personnel sujet (je, tu, il) et un substantif, les introductifs et certaines catégories, etc...

Exemple

Un verbe intransitif nécessite une préposition pour introduire chacun de ces compléments, donc la chaîne "entre les" permet de résoudre l'homographie du mot "entre" (verbe ou préposition). En effet le verbe "entrer" est intransitif, or dans la chaîne le mot "entre" est directement suivi d'un article, il ne peut donc pas être un verbe.

Dans les groupes nominaux de la forme : "article" + "substantif" ou "adjectif" + "substantif" ou "adjectif" + ..., le mot "entreprise" employé comme participe passé se place après le substantif qu'il qualifie, donc, dans les chaînes commençant par : "l'entreprise", c'est forcément un substantif.

Nous proposons d'utiliser toutes ces remarques pour résoudre certaines homographies simples, tout en sachant que nous ne pourrions pas tout résoudre, mais en remarquant quand même que ces homographies "évidentes" représentent la grande majorité des ambiguïtés rencontrées en français.

Dans le même ordre d'idées, il faut remarquer que la plupart les systèmes de résolution des ambiguïtés traitent ces ambiguïtés comme des "accidents" de la langue, c'est à dire que les ambiguïtés ne sont pas prises en charge par le modèle. A notre avis, il peut être intéressant de les traiter comme des objets apparaissant dans la modélisation de la langue. Par exemple une homographie classique sur le mot "bien", ambigu entre un adverbe et un substantif, peut être résolue en utilisant le fait qu'elle est engendrée par la chaîne de caractères "bien", que cette chaîne appartient à certains idiomes de la langue et que sa résolution ne demande pas nécessairement les mêmes traitements que toutes les ambiguïtés entre un adverbe et autre chose, ou même entre un adverbe et un substantif.

Exemple

Soit la chaîne "la réforme entreprise", "entreprise" est ambigu entre un participe passé (verbe "entreprendre") et un substantif. Un groupe nominal devant contenir un substantif (et un seul s'il n'y a pas de coordination), la catégorie "substantif" est à rejeter et nous choisissons la catégorie "participe passé", nous pouvons donc résoudre simplement cette homographie.

Une autre solution est de dire : dans le cas où une homographie entre un participe passé et un substantif apparaît après un substantif, nous choisissons la catégorie "participe passé". Nous formaliserons cela sous la forme :

"substantif" suivi de "substantif ou participe passé"
==> choix de "participe passé".

5) CONCLUSION GENERALE

Nous avons essayé de présenter, dans ce chapitre, les limitations des analyses classiques de langues naturelles appliquées à l'étude de textes libres.

L'analyse sémantique d'un texte procure un niveau de compréhension insuffisant, lié aux difficultés de modélisation et de manipulation des concepts sémantiques d'une langue naturelle. La complexité de ces analyses les rend, d'autre part, inutilisables pour des applications demandant un haut niveau de sécurité et un traitement rapide.

L'analyse syntaxique implique un temps d'exécution trop élevé pour certaines applications, notamment les systèmes interactifs. Ce temps est lié à la modélisation de la grammaire d'une langue naturelle et à la présence de nombreuses homographies, homographies de la langue, mais aussi homographies "artificielles" engendrées par l'analyse

morphologique. La non-linéarité du modèle "hors-contexte" rend peu vraisemblable une amélioration notoire, malgré l'évolution constante du matériel.

Si une adaptation de l'analyse sémantique s'avère très difficile dans le cadre d'applications très générales, l'analyse syntaxique peut, à notre avis être déchargée d'une partie de son travail :

- Le modèle hors-contexte est indispensable dans certains cas, mais un modèle d'états finis peut résoudre énormément de problèmes simples, et un pré-traitement d'états finis pourrait construire déjà une bonne partie des sous-structures d'une phrase.

- Les homographies, en créant une combinatoire importante au niveau du traitement, ralentissent énormément ces analyses. Il est possible d'en résoudre la plupart sans essayer de construire la structure des phrases complètes, mais en utilisant un traitement de surface, local à l'ambiguïté et très rapide. Ce traitement utiliserait au maximum les particularités de certains mots très importants par les constructions précises qu'ils impliquent.

Ce sont ces principes que nous avons essayé de mettre en oeuvre dans le système PIAFPS, en créant un préprocésseur à une analyse syntaxique, préprocesseur que nous avons appelé PRESYNTAXE et dont le but principal est de résoudre simplement et rapidement le plus grand nombre d'homographies. Nous avons en cela généralisé l'approche des structures de dépendances du système PIAF. Cette approche s'apparente à une démarche non grammaticale dont le seul but est d'éliminer les incohérences.

CHAPITRE 2 : PRESENTATION

A) APPLICATION ET OBJECTIFS

Ce travail a été motivé par un projet de documentation automatique. Un logiciel a été développé dans ce cadre au laboratoire IMAG; ce logiciel spécialisé (PIAFDOC : Programmes Interactifs d'Analyse du Français Appliqués à la Documentation (GRA2), (GRAV)) est une adaptation d'un logiciel plus important et très général destiné à l'analyse formelle des langues naturelles (PIAF : (COU), (VEILL1)). Quelques années d'utilisation à une échelle industrielle ont permis d'apprécier ses qualités, mais aussi de voir les limites des techniques mises en oeuvre. Ce sont ces limites que nous avons essayé de faire reculer dans ce travail.

Nous allons d'abord rappeler en quelques mots le fonctionnement des bases documentaires, puis présenter rapidement PIAFDOC et les problèmes que suscite son utilisation, pour en arriver à une description du système que nous avons réalisé.

1) GENERALITES SUR LES BASES DOCUMENTAIRES

Une base documentaire est une base de données où les objets manipulés sont des textes. Comme dans toute base de données un texte est stocké, rappelé et éventuellement édité. Dans le cadre de notre application, les documents sont caractérisés par des mots-clés, ces mots-clés sont des mots représentant les informations présentes dans le texte.

La définition d'un ensemble de mots-clés associé à un document s'appelle l'indexation.

Le rappel des documents s'effectue au moyen d'une question posée à la base documentaire. Cette question est en général une expression booléenne de mots-clés articulée autour des opérateurs classiques "et", "ou", "sauf". Un document est rappelé si les mots-clés qui lui ont été associés lors de l'indexation vérifient cette équation.

La plupart des systèmes proposent des moyens d'affiner une interrogation en interrogeant la base sur des critères secondaires. La définition de ces critères et les moyens mis en oeuvre dépendent des logiciels. Parmi ces critères, les plus courants sont des critères de positionnement relatif des mots ayant permis le rappel du document. Par exemple présence dans la même phrase de plusieurs mots, distance entre des mots inférieure à une certaine borne, etc ...

En général l'indexation est effectuée manuellement; une personne

associe à chaque document une liste de mots-clés. Cette technique pose de nombreux problèmes :

- L'indexation est une tâche fastidieuse et toute erreur durant cette phase peut avoir des conséquences importantes telles que la "disparition" d'un document. En tant qu'erreur il faut citer les erreurs d'interprétation du document et toutes les erreurs de saisie. Ces dernières sont très courantes, il suffit pour s'en convaincre de regarder un fichier inverse d'une base documentaire. D'autre part cette indexation est non-déterministe car elle est manuelle.

- Ce n'est généralement pas la même personne qui indexe les documents et qui interroge la base. Il est donc nécessaire d'adopter certaines conventions d'écriture des mots-clés, conventions communes à l'indexation et à l'interrogation. D'une façon plus générale, il faut assurer une parfaite cohérence entre ces deux opérations. Cette cohérence implique une normalisation des descripteurs.

Deux mesures caractérisent l'efficacité d'une base documentaire : le taux de rappel et le bruit.

$$\text{-taux de rappel} = \frac{\text{nombre de documents pertinents rappelés}}{\text{nombre de documents pertinents}}$$

$$\text{-bruit} = \frac{\text{nombre de documents non pertinents rappelés}}{\text{nombre de documents rappelés}}$$

Pour une meilleure utilisation de la base, il faut augmenter le taux de rappel et diminuer le bruit.

La solution pourrait être l'automatisation de la phase d'indexation. Fonctionnellement, cette automatisation aurait pour but :

1) De contrôler le texte afin d'éliminer les erreurs de saisie.

2) De repérer les mots importants présents dans le texte afin de réaliser une indexation systématique et cohérente.

3) De choisir pour ces mots un ensemble de descripteurs corrects et normalisés.

L'augmentation du taux de rappel doit être faite par une meilleure indexation des documents, d'où l'importance d'un logiciel capable d'effectuer une indexation automatique et systématique. Mais une telle automatisation passe généralement par une indexation plus complète du document. En effet, pour une sécurité d'indexation et en raison des limitations d'une analyse sémantique appliquée aux textes libres, un

logiciel d'indexation devra réaliser une analyse formelle sans interprétation de la langue. Cette analyse sera donc essentiellement morphologique et éventuellement syntaxique.

Tous les systèmes d'indexation automatique utilisent la notion de mot vide documentaire (c'est à dire un mot dépourvu de signification pour l'application concernée) et, sont retenues comme descripteurs (mots-clés), des représentations des mots non reconnus comme mots vides. Pour des raisons de sécurité cet ensemble de mots vides est assez réduit, ce qui entraine une multiplication du nombre des descripteurs.

L'obligation de rester à un niveau formel entraine une augmentation du nombre de descripteurs (et donc du bruit) pour tout gain sur le taux de rappel. Cette augmentation étant inévitable, il faut introduire des outils capables d'affiner une question et de filtrer les documents retenus, une simple interrogation sur la présence ou l'absence de mot-clés ne pouvant plus satisfaire un utilisateur (plus de 100 documents rappelés pour une dizaine de valides dans certains cas). Tout logiciel de gestion de base documentaire offre de tels outils mais ils sont généralement à la fois insuffisants et générateurs d'erreurs (élimination de documents valides). Par exemple MISTRAL (MIST) permet un affinage par des vérifications sur le positionnement relatif de chaînes de caractères dans le texte (distance, appartenance à la même phrase, au même paragraphe, ...); ces chaînes de caractères ne sont pas forcément des mots-clés mais les mots-clés sont une représentation de ces chaînes. Ces critères sont très insuffisants et, là aussi, une analyse linguistique du texte serait souhaitable. Il faudrait par exemple pouvoir utiliser des notions plus linguistiques de dépendance entre des termes, d'appartenance à des groupes syntaxiques, etc.

Exemple 1

Si un utilisateur veut les documents parlant d'un "projet de réforme foncière", deux possibilités se présentent :

- Si la base connaît le mot "projet de réforme foncière", tous les documents indexés avec lui seront rappelés.

- Si la base ne le connaît pas, l'utilisateur devra interroger avec les mots "projet", "réforme", "foncière". Seront rappelés tous les documents parlant de projet, de réforme et de foncier, mais pas forcément de "projet de réforme foncière". Il faudra affiner la question pour essayer de filtrer les documents ne contenant pas ce concept. MISTRAL propose des notions de distance entre les mots, c'est à dire que l'utilisateur pourra demander que les mots "projet", "réforme" et "foncier" soient dans la même phrase, distants de moins de n mots, etc... Il serait utile de pouvoir demander par exemple que ces trois mots soient dans le même groupe nominal, soient dépendants les uns des autres, etc...

Un logiciel a été développé au laboratoire IMAG pour résoudre ces problèmes, il s'agit de PIAFDOC que nous allons présenter brièvement.

2) PIAFDOC (GRA2),(GRAV)

Le système PIAFDOC est un préprocesseur d'énoncés écrits en langue naturelle en vue de l'utilisation de ces textes par un système de gestion de bases documentaires. Il est organisé autour d'un dictionnaire et d'une grammaire. Ces deux éléments sont des paramètres du programme et sont modifiables en cours de traitement. Ce système réalise une analyse morphologique de la langue à l'aide d'un transducteur d'états finis.

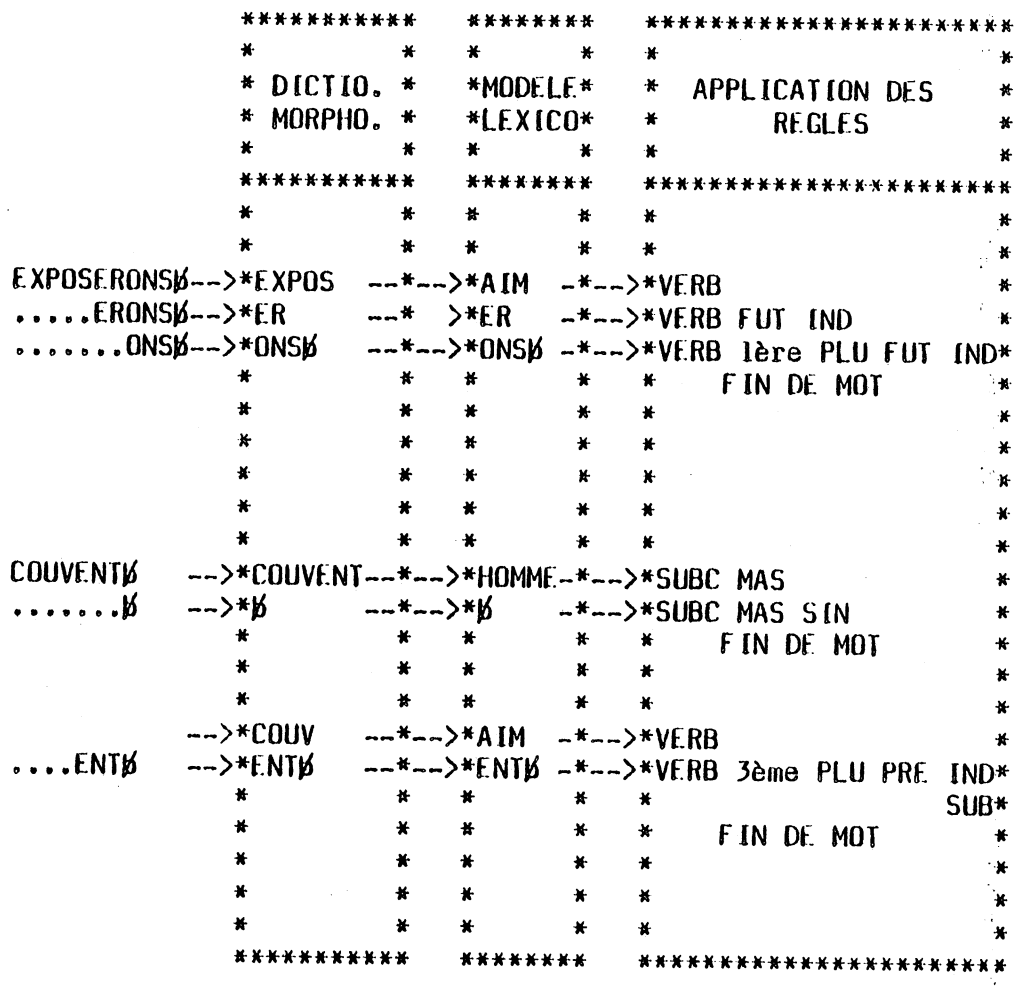
La morphologie assure :

- La segmentation du texte en "mots".
- Le contrôle de validité par rapport à un ensemble de modèles et de règles de construction des mots.
- La transduction des propriétés grammaticales (calcul des variables morphologiques).
- La recherche de toutes les solutions possibles (homographies internes et externes) : homographies syntaxiques, polysémies, ...

Le dictionnaire contient un ensemble de racines, suffixes, terminaisons, mots composés et locutions. La reconnaissance des chaînes s'effectue par identifications des plus longues sous-chaînes dans le dictionnaire. Chaque sous-chaîne fait référence à un modèle lexicographique de construction des mots. A l'aide de règles contenues dans la grammaire, le système confirme ou infirme la juxtaposition de sous-chaînes. Il faut noter que le blanc n'est pas significatif en tant que séparateur de mots.

Nous donnons ici un exemple de reconnaissance de chaînes de caractères.

Exemple 2



Les significations des termes employés sont :

Modèles
- AIM = verbes réguliers du 1er groupe
- ER = futur pour les verbes du 1er groupe
- ONSØ = 1ère personne du pluriel
- HOMME = substantifs communs masculins réguliers
- Ø = singulier pour les substantifs communs (espace)
- ENTØ = 3ème personne du pluriel pour les verbes

CL (catégories lexicales) :
- VERB = verbe
- SUBC = substantif commun

Variables : FUT = futur
IND = indicatif
lère= première personne
PLU = pluriel
MAS = masculin
SIN = singulier
PRE = présent
SUB = subjontif

La composante documentaire assure la production de mots-clés normalisés. La normalisation s'effectue par :

- L'extraction des racines pour les substantifs et les adjectifs.

- La génération de formes infinitives pour les verbes, participes passés et présents.

- L'identification de mots composés dérivés mais connexes (c'est à dire de mots composés dont les différents composants se suivent dans le texte). Par exemple la chaîne "Pollution de la mer" sera reconnue comme un mot composé (par référence au dictionnaire), mais pas la chaîne "pollution chimique de la mer". Le système est capable de reconnaître dans la chaîne "projet de réforme foncière" les deux mots composés "projet de réforme" et "réforme foncière".

- La recherche de synonymes préférentiels : le dictionnaire permet de définir des classes d'équivalence "documentaires". Pour chaque mot appartenant à une classe, le descripteur généré est un représentant préférentiel de cette classe. Par exemple les chaînes "pollution de la mer" et "pollution marine", reconnues comme "mots" par le système, appartiennent à la même classe d'équivalence et génèreront le descripteur "pollution marine".

- La décomposition de mots composés en mots simples. Une chaîne reconnue comme mot composé par la morphologie peut générer comme mot-clés l'ensemble des "mots" qui la compose. Par exemple la chaîne "réforme foncière" peut générer le descripteur "réforme foncière", mais aussi "réforme" et "foncier".

- La production de formes polysèmes avec demande d'assistance : certaines chaînes ont plusieurs significations; le système propose toutes ces significations à l'utilisateur qui choisit le descripteur. Par exemple la chaîne "concorde" signifie :

  Un avion.
  Une place.
  Un accord, une entente.

  Le système propose ces trois significations, avec les descripteurs associés à chacune d'elles, et l'utilisateur choisit le mot-clé.

Les renseignements nécessaires à cette normalisation sont stockés dans le dictionnaire. A chaque mot peut être associé un certain nombre d'indicateurs décrivant son comportement documentaire. Parmi ceux-ci on peut citer : mot vide, mot composé, possède un synonyme préférentiel, etc...

Ce logiciel a été utilisé conjointement pour assurer l'indexation de documents et pour analyser d'une façon similaire les questions posées à la base documentaire. Il considére la question comme un énoncé en langue naturelle et en extrait les mots-clés. L'utilisateur participe à la définition de l'expression booléenne équivalente à la question et cette expression est transmise à la base. Cela permet d'assurer une parfaite cohérence entre l'indexation et l'interrogation de la base par un traitement systématique et cohérent des textes et des questions : les connaissances mises en oeuvre sont les mêmes dans les deux cas.

3) LIMITATIONS

Il peut sembler étonnant de se contenter d'une analyse morphologique pour cette application. Il semble en effet que l'opération d'indexation soit avant tout une opération concernant le contenu du texte et donc sa sémantique. Mais nous avons vu dans le chapitre 1 que, dans l'état actuel des choses, il semblait assez irréaliste d'employer une analyse sémantique sur des textes libres. Les textes manipulés par PIAFDOC sont très généraux et rarement scientifiques. Ce sont, entre autres, des discours politiques, la chronologie de l'assemblée nationale, etc...

Afin de limiter les erreurs d'interprétation que tout programme "intelligent" ne manquerait pas de faire, il faut rester à un niveau formel traitable automatiquement. Une analyse sans interprétation est en effet , à notre avis, la seule solution pour parvenir aux objectifs de cette automatisation : une indexation sûre et systématique.

Un tel logiciel a besoin d'interventions humaines pour résoudre certains problèmes liés au choix des descripteurs et au contrôle de saisie. PIAFDOC est donc un programme interactif où l'utilisateur intervient pour corriger un texte, mettre à jour les paramètres internes et aider le programme à choisir certains mots-clés dans le cas d'homographies syntaxiques ou de polysémies.

C'est là que réside le principal problème de son utilisation. Actuellement le nombre d'arrêts et donc d'interventions humaines est très important (1,5 arrêts de moyenne pour des documents de 4 ou 5

lignes). La majorité de ces interruptions est due aux homographies syntaxiques et non à des problèmes sémantiques. Nous avons vu que ces homographies étaient, dans la plupart des cas, des évidences ou alors étaient générées par le modèle morphologique et n'avaient aucun sens réel. Ces nombreux arrêts nuisent énormément au logiciel : ils rendent la tâche fastidieuse aux documentalistes alors que le but du programme est justement de la leur faciliter en traitant automatiquement tout ce qui n'est pas de leur compétence.

Ces problèmes sont inhérents à l'analyse morphologique. Pour les résoudre, il faut réaliser une analyse plus fine. Nous avons considéré que tous les problèmes sémantiques étaient du ressort de l'utilisateur, donc une syntaxe est le traitement le plus complet que nous puissions réaliser et qui corresponde à nos objectifs.

Il faut remarquer que nous n'avons pas besoin de déterminer la structure des phrases, mais que seule la résolution des homographies nous intéresse. Utiliser une analyse syntaxique "classique" pour résoudre ces problèmes pénalise énormément le temps d'exécution du programme, rendant son utilisation pratiquement impossible dans un environnement interactif. Une telle analyse, comme nous l'avons remarqué précédemment, pourrait cependant être utile pour un affinage des réponses. Mais, là aussi, la présence de nombreuses homographies et sa complexité nous interdisent cette solution dans un contexte interactif.

L'affinage des questions pourrait proposer à l'utilisateur des notions de groupes syntaxiques, notions logiques et non plus uniquement formelles, comme nous l'avons montré dans l'exemple 1. Pour le faire, il faudrait délimiter dans le texte les groupes syntaxiques, par exemple les groupes nominaux et verbaux, et ceci sans avoir besoin de construire la structure complète de la phrase. Nous nous sommes intéressés à ce problème, mais il n'a pas été notre préoccupation principale. En effet, un traitement vraiment efficace n'est, à notre avis, possible qu'en utilisant une analyse syntaxique complète déchargée du repérage des constructions évidentes et de la résolution de la plupart des homographies. Un tel traitement est possible à cause des mots non ambigus qui introduisent la plupart des groupes en français (cf chapitre 1, paragraphe 4). Nous avons réalisé une maquette d'un tel traitement (module de regroupement de PIAFPS).

Le principal problème est donc de résoudre certaines homographies assez simplement, de façon à rester dans un contexte interactif rapide. Nos contraintes sont :

- Nous voulons un logiciel capable de s'exécuter sur un micro-ordinateur de taille moyenne, dans un environnement interactif (et donc avec un temps de réponse acceptable). Ceci dans le but de rapprocher l'outil informatique de l'utilisateur, car la saisie est une opération locale. Nous voulons également rester compatibles avec un logiciel déjà

existant (PIAFDOC).

- Nous voulons effectuer une analyse uniquement formelle du texte, sans interprétation sémantique. Les textes traités sont des textes très généraux et l'utilisation d'une analyse sémantique sur des textes libres ne nous parait pas réalisable. D'autre part seule une analyse sans interprétation peut donner le niveau de sécurité nécessaire à l'application documentaire.

- Le logiciel, pour des raisons de sécurité et de fiabilité, doit toujours donner un résultat correct. Il s'agit ici de la production des mots-clés, une erreur est un mauvais choix de descripteur. Si dans certains cas la validité de ce résultat ne peut être assurée, l'interaction avec un utilisateur est là pour aider le système à faire un choix.

Ces conditions empêchent donc d'utiliser une analyse syntaxique classique pour résoudre les homographies, en raison de la complexité excessive de ces méthodes, et une analyse statistique, en raison des risques d'erreurs. Par contre elles valident une analyse partielle et incomplète, l'interaction étant toujours là pour résoudre les problèmes que le système est incapable de résoudre.

4) PRESENTATION DU SYSTEME PIAFPS

4.1) Introduction

Tout d'abord, l'analyse morphologique du système PIAFDOC permet de reconnaître une chaîne de caractères comme étant un mot du langage. A cette chaîne sont associés :

- Une classe lexicale ou grammaticale (CL) décrivant la fonction syntaxique du mot. Par exemple verbe (VERB), substantif commun (SUBC), adjectif(ADJQ), participe passé (PPAS), etc...

- Certaines variables. Ce sont des compléments de la description de la fonction syntaxique du mot que nous n'avons pas jugé nécessaire de noter sous la forme de catégories syntaxiques afin de ne pas multiplier le nombre des homographies. Ces variables sont par exemple le genre (GNR), pouvant prendre les valeurs Masculin (MAS), Féminin (FEM), Neutre (NEU), le nombre (NBR) pouvant prendre les valeurs Singulier (SIN), Pluriel (PLU), etc...

- Certains indicateurs associés à la racine du mot dans le dictionnaire. Ces indicateurs décrivent le comportement documentaire du mot et peuvent contenir certaines indications sur son comportement syntaxique, indications que nous n'avons pas jugé

nécessaire de noter sous la forme de classes lexicales pour la même raison que précédemment. Par exemple certains de ces indicateurs sont : mot vide documentaire (pas de production de mot clé), synonyme préférentiel, etc... pour la documentation, et adjectif se plaçant avant le substantif, après le substantif ou n'importe où dans le groupe nominal pour le comportement syntaxique.

Ces différents niveaux de description correspondent aux différentes utilisations du programme.

La morphologie proprement dite est universelle et indépendante de toute application : les informations lexicographiques de construction des mots sont donc stockées dans la grammaire et constituent un premier niveau de description de la langue. Le traitement correspondant à ces informations permet de savoir si une chaîne de caractères est, ou n'est pas, un "mot" de la langue.

Les informations syntaxiques correspondent à un deuxième niveau de description. Elles sont calculées à partir d'informations associés au modèle dans la grammaire, et à la chaîne de caractères dans le dictionnaire. Elles dépendent des traitements ultérieurs et sont totalement arbitraires (cf chapitre 1 paragraphe 1). Ces informations sont encore assez générales et dépendent très peu de l'application.

Le troisième niveau de description est propre à l'application, ici, il correspond au comportement documentaire des mots reconnus.

Il est impossible de définir une frontière stricte entre morphologie, syntaxe et comportement documentaire. Aussi le choix de stocker une information dans un niveau particulier est totalement arbitraire. Nous avons donc permis à l'utilisateur de noter les informations syntaxiques sous plusieurs formes et dans plusieurs environnements. Nous avons défini une interface qui regroupe ces informations et qui permet de représenter un mot sous la forme d'une CL et d'attributs.

Pour les traitements un mot est donc :

- Une CL (catégorie lexicale).
- Un ensemble d'attributs, comprenant :
  - Les variables morphologiques
  - Un ensembles d'attributs regroupant les informations documentaires et toutes les autres informations disponibles (chaîne de caractères, adresse dictionnaire, etc...)

Nous dirons qu'il y a homographie quand une même chaîne de caractères possède plusieurs CL. Une homographie sera donc définie comme un ensemble de mots. Nous pourrons, si nécessaire, définir des attributs pour une homographie. Ces attributs seront définis comme la réunion des attributs des mots qui la composent.

4.2) Objectifs

Nous nous sommes donc orientés vers un mécanisme d'analyse syntaxique partielle que nous avons appelé présyntaxe.

La résolution des homographies se fera par une analyse du voisinage immédiat de l'homographie dans le texte et grâce à quelques règles de succession des catégories syntaxiques en français. Le regroupement des groupes syntaxiques utilisera des règles définissant leurs formes. Notre système devra reconnaître les homographies qu'il ne peut traiter et alors, demander une assistance à l'utilisateur.

Pour la résolution des homographies, plusieurs types de traitements se complètent. Les points communs de tous ces traitements sont :

- Résoudre une homographie, c'est trouver quelle doit être la CL associée à la chaîne de caractères reconnue par la morphologie. Nous ne nous intéressons donc qu'aux homographies syntaxiques et non aux polysémies (notre analyse est une analyse syntaxique partielle et non une analyse sémantique).

- Nous ne voulons effectuer aucun contrôle syntaxique sur le texte, c'est à dire que nous considérons que le texte d'entrée est toujours valide. Cependant, dans certains cas nous pourrons repérer certaines incohérences entre le texte et le modèle. Ces incohérences peuvent provenir des règles de résolution des homographies : l'utilisateur a donné une règle cohérente avec les autres règles mais erronée au niveau de la langue. Mais elles peuvent aussi provenir du texte lui-même. Nous en avertirons l'utilisateur qui aura la possibilité de corriger en cours de traitement l'ensemble des règles de résolution des homographies, ou le texte. Il faut en effet remarquer que les fautes de frappe sont assez courantes dans un texte. Certaines sont repérables simplement grâce à la morphologie : le mot n'existe pas. Certaines ne le sont pas : le mot existe mais est sans signification dans son contexte.

- Quand nous parlons d'environnement ou de contexte d'une homographie, nous désignons l'ensemble des mots (connus par leurs CL et leurs attributs) voisins du mot homographe dans le texte.

- Le système PIAFDOC est un système interactif; nous voulons conserver totalement cette propriété et donc permettre à l'utilisateur d'intervenir dans le traitement et de mettre à jour les paramètres du système en cours d'utilisation. De la même façon qu'un dictionnaire d'une langue naturelle n'est jamais complet, nous pensons qu'une base des connaissances de résolutions d'homographies ne peut l'être.

4.3) Mécanisme

La remarque précédente nous a amenés à définir un système où les connaissances seront représentées sous la forme de règles stockées dans une base de données que l'utilisateur pourra consulter et mettre à jour en cours de traitement. Chacune de ces règles décrira une configuration du texte où elle est applicable, c'est à dire une configuration dans laquelle l'application de la règle permet de faire un certain traitement. Nous appellerons cette configuration "contexte d'applicabilité" de la règle. Ce mécanisme est celui des systèmes de production bien connus en Intelligence Artificielle et la reconnaissance du contexte d'applicabilité d'une règle est un mécanisme de "Pattern Matching" ((NILS)). En effet toutes les règles seront du type :

condition ==> action

La partie gauche sera une suite de mots incomplètement spécifiés au niveau des attributs. La condition d'application de la règle est la présence dans le texte d'une suite de mots compatible avec cette partie gauche. Compatible signifie que la suite des CL et les attributs spécifiés doivent être identiques.

Les règles seront de plusieurs types que nous pouvons déjà diviser en deux groupes:

- Regroupement

- Résolution d'homographies.

L'action (explicite ou implicite) dépendra du type de traitement : ce sera une réécriture pour les regroupements et, soit l'élimination, soit le choix d'une CL, pour la résolution des homographies.

Les règles de regroupement signifient qu'une configuration d'entrée (partie gauche de la règle) devra être transformée en une nouvelle entité (partie droite de la règle). En pratique nous transformons un ensemble de mots du texte, c'est à dire un ensemble de couples (CL, attributs) en un nouveau "mot" dont la CL est donnée en partie droite

et les valeurs des attributs sont calculées à partir d'indications données dans la partie gauche de la règle. Ce calcul peut s'effectuer par une réunion ou une intersection des valeurs des attributs de certains des mots regroupés.

Exemple 3

Nous pouvons regrouper certains groupes verbaux très facilement, par exemple des verbes au passé composé avec une négation. Nous aurons une règle de la forme :

```
"ne" + XAV + "pas" + PPAS ==> CL       = VERB
                              Temps    = passé composé
                              nombre   = nombre de l'auxiliaire
                              personne = personne de l'auxiliaire
                              ...
```

Dans cette règle, XAV signifie "auxiliaire avoir" et PPAS "participe passé". Cette règle permet de regrouper des chaînes du type de : "n'ai pas acheté", ...

Ce mécanisme n'est pas une analyse "grammaticale" : il s'agit d'un "pattern matching" sur le texte. Si nous rencontrons une configuration présente dans une partie gauche de règle, nous effectuons la transformation, sinon nous ne faisons rien. Ce mécanisme est à opposer aux analyses syntaxiques "classiques" des grammaires génératives.

Soit la phrase : "je n'ai pas marché". Pour les besoins de l'exemple, nous admettons que la morphologie donne :

```
je      :  CL = PROS    VAR = SIN ( singulier )
n'      :  CL = NE
ai      :  CL = XAV     VAR = SIN
pas     :  CL = PAS
marché  :  CL = PPAS    VAR = MAS ( masculin ), SIN.
```

Grâce à la règle précédente, les regroupements transformerons cette segmentation en :

```
je              :  CL = PROS     VAR = SIN
n'ai pas marché :  CL = VERB     VAR = SIN
```

Par une simple identification de la chaîne des CL :

"NE XAV PAS PPAS"

Pour la résolution des homographies, il était impossible, dans un traitement local, de définir un mécanisme de validations, c'est à dire

de définir dans un certain environnement un ensemble de CL autorisées. En effet, cette technique revient à définir une grammaire générative de la langue.

Exemple 4

Si nous voulons définir quelles sont les CL qui peuvent, par exemple, apparaître après un substantif, nous devons définir :

- quelles sont les CL pouvant apparaître dans le groupe nominal après le substantif.

- quels sont les groupes pouvant suivre un groupe nominal en français, et quelles sont les CL pouvant commencer ces groupes.

Si nous répétons cette opération pour toutes les CL, nous en arrivons à décrire la structure de tous les groupes syntaxiques du français, et la façon dont tous ces groupes s'organisent dans la phrase. Nous décrivons en fait une grammaire générative du français.

Nous nous sommes orientés vers un mécanisme de filtrage : si nous ne pouvons pas définir un ensemble de CL valides, peut être pouvons nous définir un ensemble de CL interdites.

Exemple

Plutôt que d'essayer de définir les CL autorisées après un pronom personnel sujet (je, tu, il, ...), nous définirons les CL interdites : entre autres un substantif, un adjectif , ...

Cette façon de voir les choses est assez peu naturelle, mais elle présente un immense avantage: celui de pouvoir supporter un ensemble incomplet de règles. Si nous ne connaissons pas, dans une configuration donnée, l'ensemble des CL interdites, nous ne trouverons peut être pas de résultat, mais si nous en trouvons un nous pouvons en assurer la validité (à condition toutefois que les règles et le texte soient valides). Cette propriété valide la démarche, mais nous nous contentons ici de reconnaître un sur-ensemble du langage.

L'inconvénient d'un tel système est qu'il est difficile de le faire progresser par apprentissage. L'apprentissage permet de trouver un ensemble de constructions valides mais jamais un ensemble de constructions non valides. En effet, ce n'est pas parce qu'on n'aura jamais rencontré une configuration qu'elle sera interdite et nos objectifs nous interdisent de telles suppositions. Le niveau de complexité des règles n'intervient qu'au niveau des performances. Un premier mécanisme sera donc un mécanisme de filtrage et d'interdiction.

4.4) Description des règles

4.4.1) Connaissance générale

4.4.1.1) Interdictions

Ce mécanisme utilise le fait que, parfois, en français des catégories syntaxiques ne peuvent se suivre. Le fait que nous décrivions les mots par leurs CL nous a obligés à définir plusieurs types de règles d'interdiction:

- Les interdictions absolues: la succession des CL est toujours interdite. Par exemple un substantif ne peut suivre un pronom personnel sujet (je, tu, il, ...). Dans la phrase "je marche", "marche" ne peut être un substantif car cette catégorie ne peut apparaître après "je".

- Les interdictions conditionnelles: la succession n'est interdite que sous certaines conditions que nous ne pouvons décrire uniquement par des successions de CL. Ces conditions sont généralement issues du fait que la représentation d'un mot par ses CL est insuffisante. On est donc amené à définir d'une façon plus fine le contexte d'applicabilité. Pour cela nous utiliserons les attributs pour préciser un environnement. Par exemple dans la chaîne "le tranche", "tranche" ne peut être un substantif car "le" est soit un pronom personnel, soit un article, mais, en tant qu'article, il est masculin. "tranche" en tant que substantif est féminin. Les successions "le (article) tranche (substantif)", "le (pronom) tranche (substantif)" sont impossibles : la première pour l'accord en genre, la seconde par la grammaire du français. Le texte étant correct et les deux seules solutions étant "substantif" et "verbe", "tranche" est donc dans ce cas là un verbe. Il est intéressant de remarquer que nous avons pu faire ce choix sans analyser le contexte dont était issue cette chaîne.

Exemple 5

Rentrent dans cette catégorie, la plupart des règles d'accord du français, par exemple la règle d'accord en genre et en nombre entre un article et un substantif, que nous pouvons noter :

ARTD (masculin) + SUBC (féminin) ==> Interdit
Exemple : "le (article) tranche (substantif)"
...

ou alors sous une forme plus générale :

ARTD (GNR) + SUBC (GNR) ==> Interdit
ARTD (NBR) + SUBC (NBR) ==> Interdit

Les attributs "GNR" et "NBR" correspondent aux variables "genre" et "nombre" du français, l'interdiction s'applique uniquement lorsque les valeurs de ces attributs sont différentes entre les deux mots.

Pour ces deux types de règles, les actions seront l'interdiction d'une CL dans l'homographie. Elles permettent de résoudre une homographie par élimination de CL : si toutes les CL sauf une sont interdites, l'homographie est résolue car nous supposons le texte correct.

4.4.1.2) Implication

Pour aider l'utilisateur, nous avons défini une autre classe de règles: les règles d'implication. Elles demandent à une CL d'apparaître à un emplacement précis dans la configuration donnée en partie gauche de la règle. Il faut les considérer comme le complémentaire d'un ensemble de règles d'interdiction: tout ce qui n'est pas impliqué est interdit.

Ces règles ont un but limité; en effet, contrairement aux autres règles, elles "prédisent" une CL à un emplacement précis, la démarche correspondante est donc une démarche générative. Il arrive cependant en français qu'à un emplacement précis et dans un certain contexte, nous puissions prédire facilement quelle doit être la CL d'un mot. Plutôt que d'obliger l'utilisateur à écrire une règle d'interdiction pour chaque CL, nous lui donnons la possibilité de n'écrire qu'une seule règle d'implication.

Exemple

En français après la séquence "pronom personnel sujet, pronom personnel, pronom personnel" (par exemple : "je le lui") un verbe doit apparaître.

Pour formaliser cette propriété avec des règles d'interdiction, l'utilisateur devrait écrire :

"je" "le" "lui" "article" ==> Interdit.
"je" "le" "lui" "substantif" ==> Interdit.
"je" "le" "lui" "adjectif" ==> Interdit.
"je" "le" "lui" "pronom personnel" ==> Interdit.
... une règle par CL différente de la CL "verbe".

Nous lui permettons de n'écrire qu'une seule règle ayant la même

signification :

"je" "le" "lui" ==> "verbe".

Pour ces règles les actions seront d'interdire dans l'homographie toutes les CL différentes de celle impliquée.

Ces applications de règles générales devraient permettre de résoudre un certain nombre d'homographies parmi les homographies évidentes du français. En pratique ces homographies sont des homographies assez simples où les différentes CL possibles correspondent à des constructions radicalement différentes (par exemple des homographies entre un substantif et un verbe). Mais cette technique est limitée pour les homographies entre des termes voisins pouvant appartenir au même groupe syntaxique (par exemple adjectif et substantif). Nous avons donc été amenés à définir un deuxième type de traitement plus fin et plus ponctuel que le premier.

4.4.2) Connaissance pragmatique

Ce traitement repose sur des études de cas. Les règles sont du même type que précédemment: dans telle configuration (partie gauche) on choisit telle CL (partie droite). La différence avec les traitements précédents apparaît au niveau de la définition du contexte d'applicabilité. On n'indique plus une suite de CL mais une suite de CL et d'homographies. Nous définissons donc un ensemble de règles conditionnelles: elles s'appliquent quand la configuration décrite en partie gauche est présente dans le texte et sous la condition que les homographies à résoudre soient celles apparaissant, elles aussi, dans la partie gauche.

Par exemple si l'homographie sur le mot "bien" (adverbe : ADV ou substantif : SUBC) apparaît dans un texte précédée par un verbe et suivie par un article, nous pouvons choisir la catégorie "adverbe" pour "bien". C'est le cas dans la phrase : "je parle bien le français".

Nous noterons cette règle :

partie gauche : VERB + HOMOGRAPHIE("bien", ADV, SUBC) + ARTD
partie droite : ADV

La partie droite de la règle nous indique la (ou les) CL à choisir pour lever la (ou les) homographie(s) présente(s) dans la partie gauche.

Il faut se méfier de certaines règles pouvant résoudre l'homographie de cet exemple. Il est en effet très facile d'écrire des règles qui la résolvent, mais qui sont incorrectes :

VERB + SUBC ==> Interdit, est impossible à cause des verbes d'états :

"il devient secrétaire du ...", "il est nommé président du ...", etc...

Une solution peu risquée est d'écrire une règle de choix. En effet le fait de préciser l'homographie à résoudre restreint le contexte d'applicabilité de la règle et l'utilisateur voit assez facilement toute la portée de cette règle.

En cas de besoin nous pouvons imposer certaines propriétés sur les attributs afin de préciser plus finement le contexte d'applicabilité : dans un premier temps nous avons caractérisé une homographie par l'ensemble des CL qui lui sont associées, mais nous pouvons imposer une chaîne de caractères : l'homographie à résoudre sera celle générée par cette chaîne. Nous pouvons imposer certaines valeurs pour les variables morphologiques associées à chacune des possibilités de choix, etc...

Ces règles représentent un ensemble de connaissances pragmatiques. Le problème est de définir exactement l'homographie de façon à définir très exactement le contexte d'applicabilité. Nous voulons essayer, avec les règles de choix, de prendre en compte le maximum des particularités de la langue. Pour décrire ces particularités, il est nécessaire d'avoir des outils très précis et des règles d'une portée très réduite, c'est le but de ces règles de choix. En effet, écrire une règle d'interdiction peut avoir des conséquences importantes vu la généralité de ces règles et les nombreuses exceptions liées à l'emploi de certains termes en français. L'utilisateur pense à un cas particulier, écrit pour le résoudre une règle, mais l'application de cette règle dépasse de beaucoup ce cas particulier. Ces règles de choix sont donc beaucoup plus faciles à utiliser car elles ont un domaine d'application beaucoup plus limité et l'utilisateur peut ainsi plus facilement voir toutes les conséquences de cette règle.

Il faut remarquer que les traitements par interdiction restent dans le cadre d'une modélisation classique d'un langage par un mécanisme d'états finis (nous prouverons cette propriété à la fin de ce chapitre, partie B). Par contre le fait de prendre en compte les homographies comme des éléments du langage étudié nous fait sortir de cette modélisation. En effet il est impossible avec une grammaire d'états finis de noter des informations telles que la limitation du contexte d'applicabilité de la règle à l'homographie spécifiée; nous prouverons que ce mécanisme reste cependant un mécanisme d'états finis, mais sur un vocabulaire plus complexe que le vocabulaire initial.

4.5) Résumé

En résumé, les traitements sont de deux types, chacun de ces types utilisant un ensemble de règles définissant une configuration (contexte d'applicabilité) et une action à faire.

Les traitements sont:

- Regroupement
- Résolution d'homographies

Les règles sont:

- Règles de regroupement
- Règles de résolution d'homographies
  - Règles générales
    - Interdiction
    - Interdiction conditionnelle
    - Implication (complémentaire de l'interdiction)
  - Règles pragmatiques
    - Choix

Toutes les règles sont du type :

condition ==> action

La condition est la présence dans le texte d'une configuration précisée dans la partie gauche de la règle.

L'action dépend du type de traitement et de la règle, ce sera :

- Un regroupement de mots dans le cas des regroupements.
- L'élimination d'une CL pour les interdictions.
- l'élimination d'un ensemble de CL pour les implications.
- Le choix d'une CL pour les règles de choix.

4.6) Recherche et application de règles

Pour des raisons évidentes de coût, nous voulons que les recherches de règles soient guidées par le texte, c'est-à-dire que nous essayons de déterminer si une configuration du texte apparaît dans une règle et non si une règle particulière est applicable dans la configuration courante du texte. Mais le texte est une structure continue et les règles décrivent des portions de cette structure, nous sommes donc confrontés à un problème d'identification. Pour le résoudre nous avons décidé de nous immerger dans le texte là, et uniquement là, où un traitement est nécessaire. La morphologie repère les homographies et la présyntaxe analyse le voisinage de chacune d'elles.

Nous nous opposons ici aux systèmes génératifs qui analysent, avec toute la lourdeur des techniques mises en oeuvre, le texte complet et notamment les constructions très simples. Pour notre part, nous lançons les traitements uniquement quand c'est nécessaire. Les identifications de règles seront guidées par le texte, c'est-à-dire que nous disposerons d'une "fenêtre" centrée sur l'homographie et nous essaierons de trouver une règle dont la partie gauche est compatible avec le contenu de la fenêtre. Nous avons adopté ici la technique classique du "pattern matching" de l'intelligence artificielle.

Exemple 6

Soit la phrase : "je parle bien le français."

La morphologie fournit pour chacun des mots une CL et certaines variables et repère que le mot "bien" est homographe entre un substantif et un adverbe. Nous définissons initialement une fenêtre contenant le mot "bien" et étendons cette fenêtre en fonction des règles disponibles (ce mécanisme sera décrit en détail dans le chapitre 3, paragraphe 5). Si nous disposons de la règle :

VERB + HOMOGRAPHIE ("bien", ADV, SUBC) + ARTD ==> ADV

nous étendrons la fenêtre jusqu'à ce qu'elle contienne 3 mots et soit centrée sur "bien". Une identification dans la base de données des règles permettra alors de retrouver cette règle dont l'application résout l'homographie.

Toutes les règles sont indépendantes les unes des autres, c'est à dire que si nous considérons le système PIAFPS comme un système de production, ce système est commutatif. Cette indépendance signifie que, quel que soit l'ordre d'application des règles, le résultat final est le même, et que, si une règle est applicable à un instant donné, elle le restera après l'application de n'importe quelle autre règle.

Cette indépendance est due d'une part au fait que toutes les règles de résolution d'homographies s'appliquent sur le texte initial et, d'autre part, que nous n'effectuons aucune réécriture de ce texte. Interdire une CL, c'est la marquer comme interdite. Si toutes les CL, sauf une, sont marquées, l'homographie est résolue, sinon elle ne l'est pas. Il faut remarquer que les deux groupes de règles, connaissance générale et connaissance pragmatique, sont eux aussi indépendants car :

- Aucun des deux ne se sert des résultats des applications des règles de l'autre groupe : les applications se font toujours sur le texte initial.

- le texte est considéré d'une façon différente dans les deux cas : les règles de connaissance générale essaient de reconnaître des chaînes impossibles (par exemple lors du choix de la CL "substantif" pour "marche" dans la chaîne "je marche). Les règles de choix considèrent l'homographie comme un "mot" d'un langage. Pour "marche" nous ne considèrerons plus séparément "verbe" et "substantif", mais nous traiterons globalement les deux CL.

L'indépendance est aussi due au fait que nous pouvons assurer une cohérence interne de toutes les règles (paragraphe 4.7).

Cette indépendance permet d'appliquer les règles dans un ordre quelconque. Une solution serait le parallélisme entre les deux grands groupes de règles de résolution d'homographies. Pour notre part, nous avons choisi des ordres différents d'application suivant les contextes des homographies rencontrées. Pour notre application, ces choix étaient immédiats pour des raisons de coût. Mais l'application des règles pourrait être guidée par une heuristique définissant, par exemple, le groupe à appliquer en priorité, le sens d'expansion de la fenêtre et quelle CL examiner en priorité pour la recherche de règles générales.

Exemple

Pour une homographie sur le mot "bien" (adverbe ou substantif), cette heuristique nous définirait par exemple l'ordre des traitements :

- Rechercher en priorité les règles générales.

- Etendre la fenêtre au maximum vers la "gauche".

- Traiter en priorité la CL "substantif". •

Pour être efficaces ces choix devraient être basés sur des comptages. Nous pourrions par exemple, pour chaque homographie résolue, mémoriser le type de la règle qui a permis cette résolution, le sens d'expansion qui permet un nombre d'identification de règles minimal et les CL le plus facilement écartables. Un comptage sur ces valeurs nous permettrait de définir la fonction heuristique que nous venons de présenter. Ce mécanisme s'apparente à un mécanisme d'apprentissage,

mais un apprentissage ne jouant que sur la rapidité du système.

4.7) Cohérence du système

Le principal problème de ce type de système est d'assurer la cohérence des règles. En effet nous avons vu que le principe des traitements que nous avons mis en oeuvre repose sur une cohérence interne. C'est à dire que nous devons assurer que toutes les règles sont cohérentes entre elles, mais, bien sûr, nous ne pouvons assurer que les règles sont valides au niveau de la langue.

Il faut préciser que, théoriquement, la vérification de cette cohérence ne doit pas poser de problèmes car toutes les règles de résolution d'homographies s'appliquent sur le texte initial, n'effectuent aucune modification de ce texte (marquage des interdictions) et sont en nombre fini.

Nous assurons une cohérence interne d'une part des règles de regroupement et d'autre part des règles de résolution des homographies. Les regroupements effectuant une transformation du texte, nous ne pouvons assurer la validité du résultat de cette transformation car nous n'effectuons aucune analyse "grammaticale".

Les différentes contradictions possibles, et donc que nous devons repérer, sont :

- Regroupements différents : deux règles ayant la même partie gauche ont des parties droites différentes.

- Interdiction et implication : une CL interdite par une règle d'interdiction est impliquée par une règle d'implication.

- Interdiction et choix : une configuration interdite par une règle d'interdiction est générée par une règle de choix.

  Exemple

  Les deux règles :

  1) VERB + ADV + ARTD ==> Interdit (règle non valide)

  2) VERB + HOMOGRAPHIE (ADV, SUBC) + ARTD ==> ADV

  sont incohérentes car le choix dans la règle 2 de ADV a pour effet de valider la séquence "VERB + ADV + ARTD", interdite par la règle 1.

- Implication et non choix : une CL impliquée par une règle d'implication n'est pas choisie par une règle de choix dans une configuration compatible avec le contexte d'applicabilité de la règle d'implication.

  Exemple

  1) "le" implique "SUBC" (règle non valide).

  2) "je" + "le" + HOMOGRAPHIE (VERBE, SUBSTANTIF) ==> VERBE.

  Pour être cohérente avec 1, 2 devrait choisir la CL "SUBC".

- Implication et implication : deux règles n'impliquent pas la même CL dans la même configuration.

L'utilisateur pouvant rajouter des règles en cours de traitement, nous devons effectuer ces tests lors de l'ajout de chaque règle. Il suffit donc de vérifier que la règle ajoutée est cohérente avec les règles présentes dans la base (qui sont déjà cohérentes entre elles). Nous verrons en détail (chapitre 3, paragraphe 6) les moyens que nous avons mis en oeuvre pour assurer cette cohérence.

Nous allons maintenant prouver que les traitements de résolution d'homographies restent un traitement d'états finis sur un vocabulaire différent du vocabulaire "classique" d'une analyse syntaxique des langues naturelles. En effet le fait de considérer les homographies comme des éléments d'un langage nous oblige à nous placer sur un vocabulaire de parties.

B) PRESENTATION FORMELLE

Nous allons ici rappeler quelques notions de la théorie des langages. Les propriétés énoncées au début de cette étude sont très classiques, aussi nous n'en donnons pas les démonstrations.

1) GRAMMAIRES, LANGAGES ET AUTOMATES D'ETATS FINIS [AHO]

1.1) Définitions

1.1.1) Grammaires

Grammaire d'états finis

Une grammaire d'états finis G est un 4-uplet (Vt,Vn,S,P) défini par :

- Vt : Vocabulaire terminal fini.
- Vn : Vocabulaire non terminal fini tel que :
  Vt n Vn = ∅
  Vt u Vn = V
- S : Axiome avec S ∈ Vn
- P : Ensemble fini de productions de la forme :
  A -> a ou A -> aB avec A,B ∈ Vn et a ∈ Vt

Relation ==> entre deux chaînes

On dit que x ==> y (x,y ∈ Vt) si et seulement s'il existe une chaîne z ∈ Vt* et un non terminal A tel que :

x = zA
y = z a B (ou y = z a)
A -> a B (ou A -> a) étant une production de G.

On appelle **dérivation directe** une telle relation.

Relation =*=> entre deux chaînes

On dit que x =*=> y (x,y ∈ V+) s'il existe une suite finie x0,x1,...,xn (n > 0) telle que :
x0 ==> x1 ==> x2 ==> ... ==> xn avec x = x0 et y = xn

On appelle dérivation une telle relation.

Langage défini par une grammaire G

On appelle langage défini par G, que l'on note L(G), le sous-ensemble de Vt* tel que :

$$L(G) = \{ x / S =*=> x , x \in Vt^* \}$$

1.1.2) Automates

Automates d'états finis

On appelle automate d'états finis non déterministe un 5-uplet M = (Q,Vt,d,q0,F) où

- Q est un ensemble fini d'états.
- Vt est un ensemble fini de symboles d'entrée, appelé vocabulaire terminal.
- d est la fonction de transition de l'automate, c'est une fonction de Q X Vt dans P(Q) (ensemble des parties de Q).
- q0 est l'état initial de l'automate.
- F c Q : ensembles des états finaux de l'automate.

Configuration

Si M = (Q,Vt,d,q0,F) est un automate d'états finis non déterministe, on appelle configuration de M une paire (q,w) $\in$ Q x Vt*.

Une configuration de la forme (q0,w) est appelée configuration initiale.

Une configuration de la forme (q,w) avec q $\in$ F est appelée configiuration finale.

Relation ==> entre deux configurations

Soient deux configurations (q,w) et (ql,wl) d'un automate M = (Q,Vt,d,q0,F) , nous dirons que (q,w) ==> (ql,wl) si et seulement si :

$\exists$ a $\in$ Vt tel que w = a wl
et ql $\in$ d(q,a)

Relation =*=> entre deux configurations

Nous dirons que (q,w)=*=> (q',w') si et seulement s'il existe une suite a0,a1,...,an d'éléments de Vt telle que w = a0 a1 ... an w' et une suite q1,q2,...,qn d'états telles que :

(q,a0a1a2...anw') ==> (q1,a1a2...anw') ==> ... ==> (qn,anw') ==> (q',w')

Nous dirons que (q,w) =i=> (q',w') si la suite q1,q2,...,qn a une longueur i.

Langage accepté par un automate M

Soit w une chaîne de Vt* et M = (Q,Vt,d,q0,F) un automate, nous dirons que w est acceptée (ou reconnue) par M si et seulement si :

(q0,w) =*=> (q,e) avec q ∈ F.

Nous appellerons langage accepté par M l'ensemble des chaînes acceptées par M.

Automate déterministe

Soit M = (Q,Vt,d,q0,F) un automate non déterministe, nous dirons que M est déterministe si et seulement si d(q,a) n'a jamais plus d'un membre quels que soient q et a.

Si d(q,a) a exactement un membre, pour tout q et a , nous dirons que M est déterministe et complètement spécifié.

1.2) Propriétés

Théorème 1:

Soit M = (Q,Vt,d,q0,F) un automate non déterministe et L le langage reconnu par M (L = L(M)), il existe un automate M' = (Q',Vt',d',q'0,F') déterministe tel que :

L = L(M) = L(M').

Théorème 2:

Tout langage reconnu par un automate d'états finis est généré par une grammaire d'états finis.

Théorème 3:

Tout Langage défini par une grammaire d'états finis est reconnu par un automate d'états finis.

D'après les deux théorèmes précédents, nous pouvons donc dire qu'un langage est accepté par un automate d'états finis si et seulement si il est généré par une grammaire d'états finis.

Remarque

Nous venons de rappeler l'équivalence des notions de grammaire et d'automate pour la génération de langage. Par la suite, nous dirons qu'un langage est d'états finis s'il est soit généré par une grammaire d'états finis, soit reconnu par un automate d'états finis.

Théorème 4:

Le complémentaire d'un langage d'états finis est un langage d'états finis.

2) LANGAGES DEFINIS PAR INTERDICTION

2.1) Définition

Nous définissons un Langage Défini par Interdiction (LDPI) L, par la donnée du couple (Vt,I), où Vt est un vocabulaire fini et I un ensemble fini de chaînes de Vt*.

Nous dirons qu'une chaîne appartient au langage L si et seulement si elle ne contient aucune sous-chaîne contenue dans I.

$$L = \{ x \in Vt^* / \nexists w \in I \text{ et } \nexists z1, z2 \in Vt^* , z1\ w\ z2 = x \}$$

Théorème

Un LDPI est un langage d'états finis.

Démonstration

Soit L un LDPI défini par le couple (VT,I). Pour toute chaîne w = x0x1x2...xn de I nous définissons un automate Mw = (Qw,Vt,dw,qw0,Fw) avec :

Qw = {qw0,qw1,qw2,...,qwn}
Fw = {qwn}
dw(qw0,x0) = qw1
dw(qw0,x) = qw0 pour tout x ≠ x0.
dw(qwi,xi) = qwi+1 pour 1<=i<n
dw(qwn,x) = qwn pour tout x ∈ Vt.
dw non défini sinon.

Le langage reconnu par Mw peut s'écrire sous forme d'expression régulière :

Lw = L(Mw) = Vt* w Vt*

I étant fini, considérons l'automate non déterministe obtenu par "réunion" de tous les automates Mw : M = (Q,Vt,d,qo,F)

Cet automate sera défini par :

- Q Réunion de tous les Qw exceptés les états initiaux et avec le nouvel état q0.
- q0 Nouvel état.
- F Réunion de tous les états finals de tous les automates.
- d défini par :

d(q0,x) est la réunion de tous les dw(qw0,x).
d(qwi,x)=dw(qwi,x).

Cet automate reconnait le langage défini par l'expression régulière :

Vt* w1 Vt* + Vt* w2 Vt* + ... + Vt* wn Vt* où les wi sont les chaînes de I.

Par définition ce langage est le complémentaire du LDPI L, or comme il est d'états finis, le LDPI L est donc un langage d'états finis.

Remarque

La réciproque est fausse, tout langage d'états finis ne peut s'écrire sous la forme d'un LDPI.

Cela est une évidence pour les langages qui imposent à certains symboles de ne pas commencer, ou de ne pas finir, une chaîne. Soit par exemple un langage L = a(a+b)*, on semble ne pouvoir interdire à "b" de commencer une chaîne. Cet argument ne nous semble pas très important, en effet il suffit de rajouter au vocabulaire terminal deux caractères spéciaux signifiant "début de chaîne" et "fin de chaîne", auquel cas les conditions précédentes peuvent être formalisées simplement. Si nous notons "^" et "$" ces deux symboles signifiant respectivement début et fin de chaîne, l'interdiction de trouver b en début de chaîne se note "^b" dans l'ensemble I des chaînes interdites. Nous allons donner un autre exemple de langage ne pouvant pas s'écrire sous la forme d'un LDPI.

Exemple

Plaçons nous sur un vocabulaire Vt = {a,b} et considérons le langage d'états finis L dont le complémentaire Lc est défini par l'expression régulière :

Lc = a*

L est un langage où sont autorisées toutes les chaînes ne contenant pas que des a. Nous ne pouvons décrire L sous la forme d'un LDPI car pour cela il faudrait que l'ensemble des chaînes interdites I contienne toutes les chaînes composées uniquement de a. Or I doit être fini et toutes les longueurs de chaînes sont envisageables.

Les LDPI définissent donc un sous-ensemble des langages d'états finis. Il est facile de montrer que ce sous-ensemble est celui des langages dont le complémentaire peut s'écrire sous la forme d'une expression régulière du type suivant :

VT* w1 VT* + VT* w2 VT* + VT* w3 VT* + ... + VT* wn VT*

avec w1,...,wn ∈ VT*

2.1) Propriétés

Nous allons ici étudier la fermeture de l'ensemble des LDPI pour les opérations classiques de réunion, intersection, complémentation et concaténation.

Nous appellerons LI l'ensemble des langages définis ou définissables par interdiction.

Théorème 1

LI est stable pour la réunion et l'intersection.

Démonstration

Soient L1 = (Vt,I1) et L2 = (Vt,I2) deux LDPI. Considérons les ensembles Ii = I1 u I2 et Iu =I1 n I2, soient Li = (Vt,Ii) et Lu = (Vt,Iu) les deux LDPI engendrés par ces ensembles. Nous remarquons que Li = L1 n L2 et Lu = L1uL2.

La propriété est donc démontrée.

Li = L1 n L2 = (Vt,I1 u I2)
Lu = L1 u L2 = (Vt,I1 n I2)

Théorème 2

LI n'est stable ni pour la concaténation ni pour le passage au complémentaire.

Démonstration

Soit L = ({a,b},{b}) un LDPI, en termes d'expression régulière nous pouvons écrire :

L = a*

Le complémentaire de L, Lc, est un langage qui ne contiendra aucune chaîne composée uniquement de a. Nous avons vu dans l'exemple précédent qu'un tel langage ne pouvait s'écrire sous la forme d'un LDPI. LI n'est donc pas stable pour le passage au complémentaire.

Considérons les deux langages L1 et L2 définis sur un vocabulaire Vt = {a;b,c} par les expressions régulières :

L1 = (a+b)*
L2 = (a+c)*

L1 n'accepte pas les c et L2 les b. Chacun des ces langages peut s'écrire sous la forme d'un LDPI :

L1 = (Vt,{c})
L2 = (Vt,{b})

Le langage L = L1L2 obtenu par concaténation des deux langages ne peut s'écrire sous la forme d'un LDPI. Pour y parvenir il faudrait

pouvoir interdire toutes les chaînes de la forme :

ba*c

quel que soit le nombre de a, alors que les chaînes ab, ac, ba, ca sont autorisées. Or l'ensemble des chaînes interdites est un ensemble fini, cette interdiction est donc impossible.

L'ensemble LI des LDPI n'est pas stable pour la concaténation.

3) EXTENSIONS

Nous avons défini la notion de langage défini par interdiction. Nous allons maintenant présenter deux autres types de langages se ramenant à des LDPI.

3.1) Implication

Soient Vt un vocabulaire terminal fini, IM un ensemble fini de triplets (w1,a,w2) avec w1,w2 ∈ Vt* et a ∈ Vt.

Soit I l'ensemble des chaînes défini par :

$I = \{ w1\ b\ w2 \ / \ \exists (w1,a,w2) \in IM \text{ et } (w1,b,w2) \notin IM \}$

Le couple (Vt,I) défini un LDPI L. Nous dirons que L est un Langage Défini par Implication (LDPIM) et qu'il est défini par la donnée du couple (Vt,IM).

Remarque

D'une façon informelle, dans L nous obligeons certains symboles à se trouver à des emplacements précis dans certains contextes. Le triplet (w1,a,w2) signifie que si les deux sous-chaînes w1 et w2 apparaissent dans une chaîne et ne sont séparées dans cette chaîne que par un symbole, ce symbole doit être "a".

La définition de L en tant que LDPI équivaut à interdire toutes les chaînes contenant w1 et w2 séparées par un seul symbole différent de "a".

w1 et w2 peuvent être vides. Si w1 est vide, nous demandons que w2 soit toujours précédée de "a". Si w2 est vide, nous demandons que "a" suive toujours w1. Si w1 et w2 sont vides, nous ne reconnaissons que la chaîne "a".

Par définition tout LDPIM est un langage d'états finis.

Remarque

Nous avons défini le LDPIM en fonction du LDPI, donc tout LDPIM peut s'exprimer comme un LDPI. Par contre la réciproque est fausse : tout Langage Défini Par Interdiction ne peut pas s'exprimer comme un Langage Défini Par IMplication.

Exemple

Soient VT = { a,b,c } un vocabulaire terminal et I = { ab } un ensemble définissant un LDPI L. Ce langage L ne peut pas s'exprimer sous la forme d'un LDPIM, car interdire à un "b" d'apparaître après un "a" n'est pas formalisable en tant qu'implication. En effet les deux chaînes "aa" et "ac" sont autorisées.

3.2) Choix

Soit Vt un vocabulaire terminal, nous définissons les ensembles V1 et V2 par :

V1 = P(Vt) - ∅ (ensemble des parties de Vt sauf la partie vide).
V2 = { (p,a) ∈ V1 x Vt / a ∈ p }

Soit C un sous-ensemble fini de (V2 u Vt)*, nous définissons un langage L, que nous appellerons langage défini par des choix (LDPC), par la donnée du couple (Vt,C) et par les règles suivantes :

Soit Ca l'ensemble de chaînes de V2* obtenues par remplacement dans chaque chaîne de C de chaque symbole a de Vt par les couples (p,a) de V2. Nous appellerons ces chaînes "chaînes autorisées". Soit Ci l'ensemble des chaînes de V2* obtenues par remplacement dans les chaînes de Ca de chaque couple initial (qui y était dans C) (p,a) par les couples (p,b) avec b ≠ a.

Le LDPC L = (Vt,C) est défini comme le LDPI (V2,Ci).

Remarque

D'après la définition précédente, tout LDPC est un langage d'états finis.

Les LDPC formalisent la notion de choix, une règle de choix est une suite de CL et d'homographies, c'est-à-dire une suite de symboles et d'ensembles. Nous symbolisons le choix proprement dit par un couple (p,a) où "p" représente l'ensemble de tous les choix possibles et "a" le symbole à choisir.

Les règles de choix sont représentées par des suites de symboles et de couples. Nous nous placerons donc sur un vocabulaire de couples (p,a). Nous transformons chaque CL apparaissant dans la "règle" initiale en des éléments de ce vocabulaire : une CL "b" peut toujours être considérée comme le résultat d'un choix et cela de plusieurs façons possibles. Nous remplaçons donc chaque symbole du vocabulaire initial apparaissant dans une règle par un couple dont il est le deuxième élément. Il faut générer toutes les possibilités pour chaque symbole. Ces remplacements définissent l'ensemble des chaînes autorisées. Pour nous ramener à un langage défini par interdiction, nous créons un ensemble de chaînes interdites par remplacement des choix donnés par la règle (c'est à dire des couples (p,a) présents dans la règle initiale) par les "mauvais" choix (c'est à dire par les couples (p,b) avec b $\neq$ a).

Par exemple si nous avons une règle de choix :

VERB + [ADV,SUBC] + ARTD ==> ADV

Nous commençons par formaliser le choix en créant un couple ({ADV,SUBC},ADV) le symbolisant. la règle s'écrit maintenant :

VERB + ({ADV,SUBC},ADV) + ARTD

Nous unifions le vocabulaire et pour cela nous transformons "VERB" et "ARTD" par des choix. La CL "VERB" sera donc transformée en tous les couples où elle peut apparaitre en partie droite, c'est à dire :

({VERB},VERB)
({VERB,SUBC},VERB)
({VERB,ADJQ},VERB)
({VERB,SUBC,ADJQ},VERB)
...

Nous faisons la même chose avec la CL "ARTD", et effectuons les produits en croix pour revenir à des chaînes du vocabulaire de couples. Les chaînes autorisées seront donc :

({VERB},VERB) ({ADV,SUBC},ADV) ({ARTD},ARTD)
({VERB,SUBC},VERB) ({ADV,SUBC},ADV) ({ARTD},ARTD)
...

Nous obtiendrons les chaînes interdites en remplaçant le couple ({ADV,SUBC},ADV) par le couple ({ADV,SUBC},SUBC).

Exemple:

Soit Vt = { a,b }

Nous définissons V2 = { ({a,b},a),({a,b},b),({a},a),({b},b) } Pour la suite et, pour des raisons de lisibilité, nous écrirons ab/a le couple ({a,b},a) et noterons "+" la concaténation de symboles. Donc V2 = { ab/a, ab/b, a/a, b/b }.

Soit par exemple C = {a+ab/a+b }. Pour obtenir Ca, nous allons remplacer "a" et "b" par les couples :

pour "a" : a/a
ab/a

pour "b" : b/b
ab/b

Ca sera donc l'ensemble :

Ca = { a/a+ab/a+ab/b , a/a+ab/a+b/b , ab/a+ab/a+ab/b , ab/a+ab/a+b/b }

Pour obtenir l'ensemble des chaînes interdites Ci, nous remplaçons le choix initial : ab/a par les autres possibilités de choix, c'est à dire : ab/b. Ci sera donc l'ensemble :

Ci = { a/a+ab/b+b/b, a/a+ab/b+ab/a, ab/a+ab/b+b/b, ab/a+ab/b+ab/b }

Le LDPC (Vt, C) est donc un LDPI, le LDPI défini par (V2,Ci).

Remarque

Tout LDPI défini sur un vocabulaire Vt "induit" un LDPI sur V2.

Soit L un LDPI défini par le couple (Vt,I), considérons Il l'ensemble de chaînes de V2* définies par remplacement dans chaque chaîne de I, de chaque symbole a de Vt par les couples (p,a) de V2. Par construction, le couple (V2,Il) défini un LDPI.

Tout LDPIM défini sur un vocabulaire Vt "induit" un LDPI sur V2.

Nous avons vu que tout LDPIM L = (Vt,IM) pouvait s'écrire sous la forme d'un LDPI Ll = (Vt,I). Le langage induit de L sur V2 sera donc le langage induit de Ll sur V2.

4) CONCLUSION

Les définitions de ce chapitre nous amènent à réunir les trois notions de langages définis par interdiction, implication et choix. Nous appellerons langage défini par interdiction, implication et choix (LDPIIMC) tout langage défini par un 4-uplet (Vt,I,IM,C) où

- Vt est un vocabulaire terminal fini.
- I est un sous ensemble fini de Vt*.
- IM est un ensemble fini de triplets (w1,a,w2) avec w1,w2 ∈ Vt* et a ∈ Vt.
- C est un sous-ensemble fini de (V2 ∪ Vt)* où V2 est défini par

$$V2 = \{ (p,a) \;/\; p \in P(Vt)-\emptyset \text{ et } a \in p \}$$

Le langage L sera défini sur le vocabulaire V2 par l'intersection des langages :

L1 : Langage induit sur V2 par le LDPI (Vt,I).
L2 : Langage induit sur V2 par le LDPIM (Vt,IM).
L3 : Langage défini par des choix (Vt,C).

Ce langage L peut s'exprimer comme un LDPI sur le vocabulaire de couples V2 (l'ensemble des LDPI est stable pour l'intersection). C'est donc un langage d'états finis sur ce vocabulaire.

CHAPITRE 3 : DESCRIPTION DU SYSTEME PIAFPS

Nous avons réalisé un prototype d'un traitement présyntaxique de textes écrits en langue naturelle afin de résoudre certaines homographies. Nous avons adjoint ce prototype au logiciel PIAFDOC, le rebaptisant PIAFPS. Ce travail a été réalisé sur un IRIS80 et nos programmes sont écrits en LP80.

1) PARTICULARITES DE L'IMPLANTATION

Nous l'avons dit, le but du système PIAFPS est d'aider un utilisateur lors de l'indexation de documents. Nous n'essaierons donc de résoudre des homographies que si elles sont importantes au niveau de la génération de descripteurs. En pratique, nous ne résoudrons pas une homographie quand :

- Tous les mots qui la composent ont l'indicateur "mot vide documentaire" (aucune production de mot-clé).

- tous les mots-clés engendrés sont identiques (même chaîne de caractères) : par exemple "Français" est ambigu entre un substantif et un adjectif. Dans les deux cas le mot-clé généré est "français". Nous ne chercherons donc pas à résoudre cette homographie syntaxique.

Remarque

Ce type d'optimisation locale est une des propriétés des mécanismes que nous mettons en oeuvre. Nous effectuons un traitement local et partiel, ce qui permet d'activer les traitements uniquement quand c'est nécessaire.

Il peut arriver, et nous le verrons en détail (paragraphe 5.2.2), que pour résoudre une homographie "utile", nous soyons obligé d'essayer d'en résoudre une qui, a priori, ne nous intéressait pas, cela lors des homographies multiples.

le module de regroupement ne sert ici qu'à accélérer les traitements en réduisant le nombre de mots de la phrase; il permet également de résoudre simplement et très rapidement certaines homographies évidentes. Il nous a semblé dommage de reporter à un traitement ultérieur ce que nous pouvions réaliser ici. Par exemple, le mot "été" est toujours ambigu entre le participe passé de l'auxiliaire "être" et le substantif; toutes les configurations où il est précédé de l'auxiliaire "avoir" nous permettent de choisir le participe passé. Les regroupements parcourant le texte de gauche à droite, il est inutile

de revenir sur cette homographie par la suite.

C'est pour cette raison que nous avons conservé ces traitements qui ne sont pas indispensables au bon fonctionnement du système.

Les attributs que nous utilisons pour la résolution des homographies sont les deux variables morphologiques "GENRE" et "NOMBRE". Un mot est donc caractérisé par :

- Une CL.
- Une, plusieurs ou aucune valeur pour les variables "genre" et "nombre".

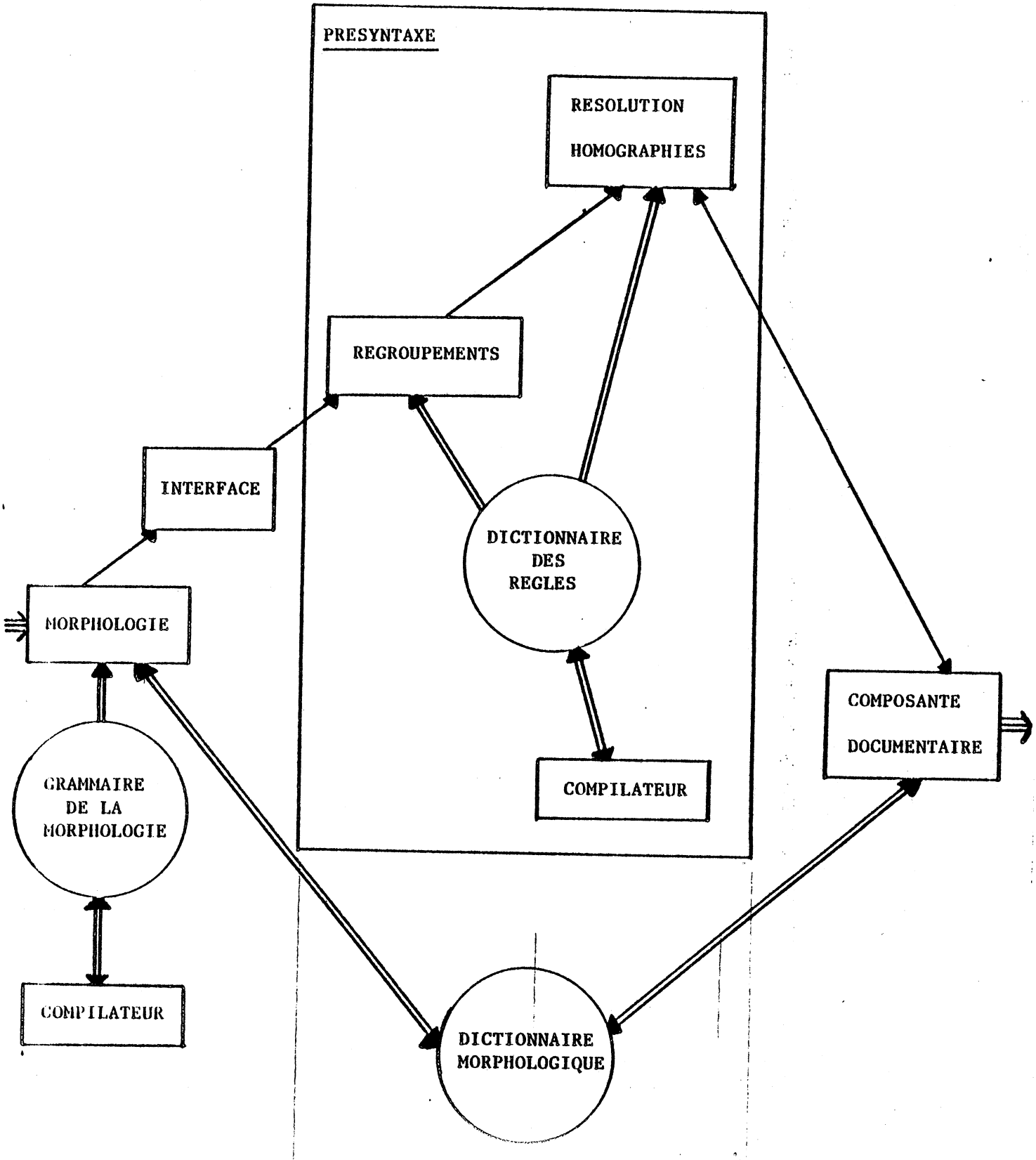
2) ORGANISATION GENERALE

Le système PIAFPS se décompose en quatre modules de traitements, ce sont :

- La morphologie
- Les regroupements
- La résolution des homographies
- La génération de mots-clés

Nous donnons ici un schéma général du système.

SCHEMA GENERAL DE PIAFPS



La morphologie étant universelle et indépendante de toute application, il est nécessaire de définir une interface avec la présyntaxe. Cette interface effectue une réécriture et un filtrage des informations fournies par la morphologie. Seuls les renseignements utiles à la présyntaxe sont conservés. Les règles de filtrage et de réécriture sont définies par l'utilisateur et stockées dans la grammaire de transformation. Nous donnerons dans l'annexe 1 une description détaillée de cette interface.

Ces traitements sont organisés de façon séquentielle : le module de contrôle est le module de résolution des homographies, c'est lui qui commande les lectures de texte (via les regroupements et la morphologie), les écritures(via la composante documentaire) et la mise à jour de la base de données des règles (via un compilateur incrémentiel).

La morphologie et la composante documentaire sont celles, à peu de chose près, du système PIAFDOC ((GRA2), (GRAV)) et nous ne les décrirons pas en détail ici. Nous avons rajouté quelques fonctions à ces modules, pour, entre autre, accepter des textes écrits en typographie riche (voyelles accentuées) et accepter des indicateurs présyntaxiques dans la description des mots à insérer dans le dictionnaire : par exemple l'indicateur "se place avant le substantif" pour un adjectif. Ces indicateurs sont utilisés lors de la réécriture des résultats de la morphologie par l'interface précédente.

Les regroupements utilisent le même compilateur de règles que la résolution des homographies; les règles de regroupement sont stockées dans la même base que les règles de résolution des homographies.

3) GENERALITES SUR LA RECHERCHE DES REGLES

Nous voulons nous immerger dans le texte dans le voisinage des homographies que nous voulons résoudre et nous voulons être guidés par le texte lors de la recherche des règles applicables.

Pour résoudre ces problèmes, nous avons défini un mécanisme de règles partielles qui permettent à tout instant de savoir si, dans la configuration courante du texte, il existe des règles qui pourront éventuellemnet être appliquées, c'est-à-dire si cette configuration courante apparaît dans la partie gauche d'une règle.

Nous verrons en détail les algorithmes d'application et de

recherche de règles dans les paragraphes 4.2 et 5.2. D'une façon générale, nous disposons d'une fenêtre contenant le groupe à regrouper ou l'homographie à résoudre. L'expansion de cette fenêtre sera guidée par les règles partielles. Si il existe une règle partielle coincidant avec la fenêtre, il peut exister des règles applicables, il faut donc étendre la fenêtre dans le(s) sens donné(s) par cette règle. Si il n'existe aucune règle partielle compatible avec la fenêtre, il n'existe sûrement pas de règles applicables.

Pour chaque nouvelle règle insérée dans le dictionnaire, le compilateur génère des règles partielles. Pour les regroupements, le texte étant examiné de gauche à droite, ne seront générées que les partielles commençant par le premier mot apparaissant dans la règle. Pour la résolution des homographies, l'expansion des configurations examinées se faisant dans les deux sens, toutes les partielles seront générées.

Deux indicateurs associés à la règle indiquent le sens dans lequel il faut étendre la fenêtre pour retrouver la règle initiale :

">" signifie vers la droite.
"<" signifie vers la gauche.

Exemple

La règle de regroupement :

/ NE + XAV + PAS + PPAS / R / VERB /

génèrera les règles partielles :

/ NE (>) / R /
/ NE + XAV (>) / R /
/ NE + XAV + PAS (>) / R /

La règle de choix :

/ VERB + (ADV , SUBC) + ARTD / C / ADV /

génèrera les partielles :

/ VERB (>) / C /
/ VERB + (ADV , SUBC) (>) / C /
/ (ADV , SUBC) (<)(>) / C /
/ (ADV , SUBC) + ARTD (<) / C /
/ ARTD (<) / C /

Dans ces règles partielles le deuxième champ de la règle (/ C / ou / R / pour l'exemple précédent) indique le type de la règle dont est issue la partielle :

/ R / signifie règle de regroupement.
/ C / signifie règle de choix.
/ I / signifie règle d'interdiction ou d'accord de variables.
/ O / signifie règle d'implication.

Si le dictionnaire des règles contient déjà une règle partielle identique, correspondant au même traitement (même deuxième champ), nous ne générons pas de nouvelle règle mais complétons, si nécessaire, les indicateurs de sens.

Exemple

Soient les deux règles de choix :

1) / NE + (VERB , SUBC) + PAS / C / VERB /
Si une homographie entre un substantif et un verbe apparaît entre "ne" et "pas" on choisit le verbe.

2) / PROS + (VERB , SUBC) / C / VERB /
Si une homographie entre un substantif et un verbe apparaît après un pronom personnel sujet (je, tu, il,...) on choisit le verbe.

Les règles partielles susceptibles d'être générées sont :

Pour 1 : / NE (>) / C /
/ NE + (VERB,SUBC) (>) / C /
/ (VERB , SUBC) (>)(<) / C /
/ (VERB , SUBC) + PAS (<) / C /
/ PAS (<) / C /

Pour 2 : / PROS (>) / C /
/ (VERB , SUBC) (<) / C /

En fait le dictionnaire contiendra les règles partielles :

/ NE (>) / C /
/ NE + (VERB,SUBC) (>) / C /
/ (VERB , SUBC) (>)(<) / C /
/ (VERB , SUBC) + PAS (<) / C /
/ PAS (<) / C /
/ PROS (>) / C /

Les règles partielles indiquent uniquement une éventuelle présence de règles globales; pour cela nous n'avons pas besoin de noter de quelle règle globale est issue chaque partielle. Cette propriété permet la factorisation des règles partielles.

Remarque

A partir d'une règle globale nous sommes obligés de générer toutes les règles partielles possibles car nous ignorons quelle va être la CL ou l'homographie par laquelle nous allons commencer les traitements :

Si nous disposons de la règle : / PROS + SUBC / I / générant les règles partielles :

/ PROS (>) / I /
/ SUBC (<) / I /

signifiant qu'un substantif ne peut suivre un pronom personnel sujet, cette règle servira (éventuellement) pour résoudre une homographie entre un substantif et d'autres CL. Dans ce cas nous accéderons à cette règle par la CL "SUBC". Mais elle servira aussi pour résoudre une homographie entre un pronom personnel sujet et autre chose (par exemple une homographie sur "tu", pronom personnel ou participe passé du verbe "taire") auquel cas nous accèderons à cette règle par la CL "PROS". En systématisant la génération des règles partielles, nous générons sans doute trop de règles (certaines CL n'apparaissent jamais dans des homographies). Mais le repérage de ces CL "non homographes" pose de nombreux problèmes pour un gain minime. Parmi ceux-ci on peut citer un problème de cohérence et d'évolution des connaissances du programme : si une CL est "non homographe", est-ce parce qu'effectivement elle ne l'est pas dans la langue ou parce que le dictionnaire de la langue dont dispose le programme est incomplet et ne contient pas des mots homographes entre cette CL et une autre ? Le choix des CL et de la classification des mots étant arbitraire, nous ne pouvons prendre en compte ces propriétés.

4) REGROUPEMENT

4.1) Description des règles de regroupement

Chaque règle comporte trois champs :

- Une partie gauche servant à l'identification (image du texte).
- Un deuxième champ indiquant le type de la règle (ici règle de regroupement).
- Une partie droite contenant la CL générée par l'application de la règle.

La partie gauche comprend deux parties optionnelles :

- Un indicateur de règle (S) : il indique que la règle reste applicable même si une des CL de la partie gauche appartient à un mot homographe. Auquel cas, l'homographie est résolue en choisissant la CL apparaissant dans la règle.

- Le marqueur de variables (£) : il indique qu'il faudra prendre en compte les variables du mot dont la CL est marquée pour le calcul des variables du mot généré par l'application de la règle (la méthode de calcul des nouvelles variables est décrite au paragraphe 4.3).

Exemple:

/NE + VERB(£) + PAS (S)/ R / VERB /

Les variables du mot généré seront les variables du mot du texte dont la CL est "VERB". Cette règle s'applique même si l'un des mots dont la CL apparaît en partie gauche est homographe (indicateur (S)). Par exemple dans la chaîne "ne marche pas", "marche" est sûrement un verbe.

/NE(£) + VERB (£) + PAS (S) / R / VERB /

Par rapport à la règle précédente, il faudra aussi prendre en compte les variables du mot dont la CL est "NE" pour calculer les variables du mot généré.

4.2) Algorithme

Ce module évalue le texte de gauche à droite. Il examine la chaîne des CL, image du texte, fournie par la morphologie. Il procède en deux phases :

- Identification des règles applicables
- Application des règles

Les identifications des règles sont dirigées par le texte. Nous disposons d'une "fenêtre" sur le texte. Au cours d'un essai de regroupement cette fenêtre s'étend vers la droite en fonction de la présence de règles partielles compatibles avec son contenu.

Nous supposons, pour l'écriture de cet algorithme, que nous disposons des primitives :

- IDENTIFIE (FENETRE) : cette primitive effectue une identification dans le dictionnaire des règles afin de retrouver les règles partielles compatibles avec la fenêtre passée en paramètre. Elle positionne les logiques :

  "GLOBAL" : "vrai" s'il existe une règle globale compatible avec la fenêtre, "faux" sinon.

  "PARTIEL" : "vrai" s'il existe une règle partielle compatible avec la fenêtre, "faux" sinon.

- PREMIER (MOT) : fonction délivrant la première CL du mot passé en paramètre.

- SUIVANT (MOT , CL_COURANTE) : fonction délivrant la CL suivant "CL_COURANTE" du mot passé en paramètre. Elle prend la valeur "NIL" une fois que toutes les CL ont été examinées.

- MEMORISER : mémorise la (les) règle(s) globale(s) repérée(s) par "IDENTIFIE".

- MOT_SUIVANT (FENETRE) : fonction délivrant le mot suivant de la fenêtre passée en paramètre.

- Nous noterons "!!" la concaténation de chaînes.

Algorithme 1

```
PROCEDURE REGROUPE ( FENETRE, MOT_COURANT )
VARIABLES : CL_COURANTE ;
            MOT_SUIV ;

DEBUT

   IDENTIFIE ( FENETRE );

   SI GLOBAL ALORS MEMORISER ;

   SI PARTIEL ALORS
   DEBUT

      MOT_SUIV := MOT_SUIVANT ( FENETRE );
      CL_COURANTE := PREMIER ( MOT_SUIV );

      TANTQUE CL_COURANTE <> NIL FAIRE
      DEBUT

         REGROUPE ( FENETRE !! CL_COURANTE, MOT_SUIV );
         CL_COURANTE := SUIVANT ( MOT_SUIV , CL_COURANTE );

      FIN;

   FIN;

FIN;
```

Exemple:

Soit le texte : "bien mangé", la morphologie donne :

```
"bien"   : CL = ADV        et        CL = SUBC
           VAR = ...                 VAR = MAS, SIN
mangé    : CL = PPAS
           VAR = ...
```

Si nous disposons de la règle : / ADV + PPAS / R / PPAS /
et donc de la règle partielle : / ADV (>) / R /
La séquence d'identification sera :

```
/ ADV /          ---> On trouve / ADV (>) / R /          règle partielle.
/ ADV + PPAS /   ---> On trouve / ADV + PPAS / R / PPAS / règle globale.
/ SUBC /         ---> On ne trouve rien.
```

Remarque

Nous lançons toujours les identifications à partir de la fenêtre courante. Nous ne tenons donc pas compte du cheminement de l'algorithme pour arriver à cette fenêtre. Dans l'exemple précédent lors de l'identification de "ADV + PPAS", nous "oublions" que nous avons déjà identifié "ADV" dans le dictionnaire. Les règles étant stockées sous forme de chaînes de caractères (paragraphe 6.3), cette information pourrait accélérer les recherches dans le dictionnaire de règles. Nous n'avons pas réalisé cette optimisation pour une raison d'uniformisation des structures de données et des traitements : cette optimisation n'était pratiquement utilisable que pour les regroupements (la résolution des homographies demande une expansion de la fenêtre vers la gauche et la droite). Le fait de lancer les identifications sur une chaîne nous permet d'uniformiser tous les accès que nous affectuons dans les dictionnaires (morphologie et dictionnaire de règles). En effet les données sont stockées sous la forme de chaînes de caractères dans des dictionnaires du type de celui manipulé par la morphologie.

D'autre part, le nombre réduit d'éléments du dictionnaire de règles ne nous semble pas justifier une telle optimisation. A notre avis les gains dus à l'uniformisation des traitements sont plus intéressants.

Nous disposons donc à la fin de la phase d'identification d'un ensemble de règles applicables.

4.3) Application des règles

Le but principal de cette étude est la résolution des homographies. Pour cette raison et pour éliminer les problèmes de cohérence que soulevait ces regroupements, nous avons axé l'application des règles vers la sécurité d'emploi : chaque fois qu'il y a un risque de conflit entre plusieurs règles, nous n'effectuons aucun traitement. Une règle est donc appliquée uniquement dans les cas suivants :

- Pas d'homographie dans les mots concernés et une seule règle applicable.

- Homographie et une seule règle applicable avec l'indicateur (S).

L'application d'une règle génère un nouveau "mot" dont la CL est la CL indiquée en partie droite de la règle et les variables sont calculées à partir des variables des mots marqués par le marqueur de variables.

PIAFDOC hiérarchise la notion de variables en définissant des variables (par exemple le genre et le nombre) et des valeurs pour chacune de ces variables (par exemple masculin et féminin pour le genre, singulier et pluriel pour le nombre). Nous avons conservé ces deux niveaux pour le calcul des nouvelles variables.

Pour les mots marqués par le marqueur de variable, nous effectuons une intersection des valeurs des variables communes et une union des valeurs des autres variables.

Exemple

Supposons que nous disposions des variables "NOMBRE" pouvant prendre les valeurs "SINGULIER" et "PLURIEL", et "NEGATION" pouvant prendre les valeurs "VRAI", "FAUX" et "INCONNU". La valeur "INCONNU" signifie que la variable "NEGATION" est inexistante pour le mot considéré.

La règle : / NE (£) + VERB (£) + PAS / R / VERB / appliquée à la chaîne : " n'achète pas ", avec :

"n'" : CL = NE , VARIABLES = NEGATION (VRAI).
"achète" : CL = VERB, VARIABLES = NOMBRE (SINGULIER)
NEGATION (INCONNU)
"pas" : CL = PAS , VARIABLES = ... (inutilisées)

permet de construire le mot :

"n'achète pas" : CL = VERB , VARIABLES = NOMBRE (SINGULIER)
NEGATION (VRAI)

On relance un nouveau regroupement à partir du dernier regroupement effectué dans le cas d'une application de règle, et à partir du mot suivant dans le cas où aucune règle n'a été appliquée.

Exemple:

Soit la phrase : "j'ai été bien ..."

La morphologie donne :

j' : CL = PROS, VAR = SIN
ai : CL = XAV, VAR = SIN
été : CL = PXET et CL = SUBC
bien : CL = ADV et CL = SUBC

Les règles globales de regroupement sont :

/ XAV(£) + PXET (S) / R / XET /
/ XET(£) + ADV / R / XET /

Le traitement sera :

1) identification:"PROS",on ne trouve rien.

On transmet donc le mot : CL = PROS, VAR = SIN

2) identification : "XAV", on trouve : /XAV(>)/R/
" :"XAV + PXET", on trouve :/XAV(£)+PXET(S)/R/XET/
" : "XAV + SUBC", on ne trouve rien.

La première règle ayant l'indicateur (S), on l'applique. On génère donc un mot avec : CL = XET et VAR = SIN.

3) identification : "XET"(mot généré), on trouve / XET (>) / R /
" : "XET+ADV", on trouve /XET(£) + ADV / R /XET /
" : "XET + SUBC", on ne trouve rien.

La règle trouvée n'ayant pas l'indicateur (S) et un des mots concernés étant homographe on n'applique rien.

4) identification : "ADV", on ne trouve rien.
" : "SUBC", on ne trouve rien.

On transmet donc l'homographie, sans effectuer de regroupements sur le mot "bien".

Nous obtenons donc :

j' : CL = PROS
ai été : CL = XET, VAR = SIN.
bien : CL = ADV et CL = SUBC.

5) RESOLUTION DES HOMOGRAPHIES

5.1) Description des règles de résolution des homographies

Nous disposons de trois types de règles. Chacune de ces règles comprend :

- Une partie gauche servant à l'identification (image du texte)
- Un deuxième champ décrivant le type de la règle et le traitement à faire (Interdiction, Implication ou Choix).
- Eventuellement une partie droite dans le cas d'une règle de choix.

5.1.1)Règles d'interdiction et d'accord de variables

Ces règles ont deux fonctions différentes suivant la présence ou l'absence de variables. Dans tous les cas elles indiquent que, dans la configuration où elles s'appliquent, la succession dans le texte de mots dont les CL apparaissent dans la partie gauche de la règle est interdite.

Si les variables sont présentes, l'interdiction s'applique quand il n'y a pas accord sur la variable spécifiée ou quand on rencontre la configuration précisée dans le cas de valeurs de variables explicites.

Si les variables sont absentes, l'interdiction s'applique toujours.

Exemple:

/ ARTD (SIN) + SUBC (PLU) / I /

Un substantif au pluriel ne peut suivre un article au singulier (cas particulier de la règle d'accord en nombre entre l'article et le substantif).

/ ARTD (GNR) + SUBC (GNR) / I /

Accord en nombre entre l'article et le substantif. Cette règle interdit le choix de l'une de ces CL quand ce choix entraine une non-vérification de cet accord.

/ ADJQ (GNR) + SUBC (GNR) / I /

Accord en genre entre un adjectif qualificatif et un substantif. L'interdiction s'applique quand cet accord n'est pas vérifié.

/ PROS + SUBC / 1 /
Un substantif commun ne peut suivre un promnom personnel sujet.

5.1.2) Règles d'implication

Nous demandons qu'une CL apparaisse à un endroit précis de la configuration décrite par la partie gauche de la règle. Cet emplacement est précisé par le caractère ">", précédant la CL obligatoire.

Exemple:

/ PROS + APRO + APRO > VERB / 0 /

La configuration : pronom personnel sujet suivi de deux pronoms personnels objets, implique la présence d'un verbe comme mot suivant. Par exemple la chaîne "je le lui" doit être suivie par un verbe.

Remarque:

Ces règles apparaissent comme complémentaires des règles d'interdiction (au moins pour les règles d'interdiction non conditionnelles); on peut, en effet, comprendre une règle d'implication comme le complémentaire d'un ensemble de règles d'interdiction : tout ce qui n'est pas impliqué par cette règle est interdit.

5.1.3) Règles de choix contextuel

Contrairement aux deux autres types de règles, ces règles décrivent l'homographie sur laquelle elles peuvent s'appliquer en plus du contexte d'applicabilité. La partie gauche contient entre parenthèses l'ENSEMBLE des CL de l'homographie , la partie droite contient les CL qu'il faut choisir quand on rencontre la configuration décrite par la règle.

Exemple:

/ PROS + XAV + (PXET , SUBC) + ADJQ / C / PXET /

PROS = pronom personnel sujet (je, tu, il ,...)
XAV = auxiliaire avoir
PXET = participe passé de l'auxiliaire être
ADJQ = adjectif qualificatif

Cette règle permet de résoudre l'homographie sur le mot "été" (substantif ou participe passé de l'auxiliaire "être") en choisissant le participe passé de l'auxiliaire être quand :

- les seules CL de l'homographie sont PXET et SUBC (L'analyse présyntaxique connaît les homographies uniquement par leur ensemble de CL).

- on rencontre cette homographie dans la configuration précisée par la règle : précédée par un pronom personnel sujet et l'auxiliaire avoir, et suivie par un adjectif qualificatif.

Par exemple dans la phrase : "J'ai été heureux de faire cela".

5.1.4) Règles partielles

Le compilateur génère automatiquement pour chacune de ces règles un ensemble de règles partielles dont le format est :

/ PG (>) (<) / TYPE /

La partie gauche (PG) est une sous-chaîne de la partie gauche de la règle initiale. Les indicateurs (>) et (<) signifient que ces règles sont des règles partielles et donnent le(s) sens dans le(s)quel(s) il faut se déplacer pour pouvoir retrouver la règle globale.

(<) signifie vers la gauche
(>) signifie vers la droite

Exemple:

/ PROS (NBR) + VERB(NBR)/ I /

génère les règles :

/ PROS (>) / I /
/ VERB (<) / I /

/ PROS + XAV + (PXET , SUBC) + ADJQ / C / PXET /

génère les règles partielles :

/ PROS (>) / C /
/ PROS + XAV (>) / C /
/ PROS + XAV + (PXET,SUBC) (>) / C /
/ XAV (>)(<) / C /
/ XAV + (PXET , SUBC) (<)(>) / C /
/ XAV + (PXET , SUBC) + ADJQ (<) / C /
/ (PXET , SUBC) (<)(>) / C /
/ (PXET , SUBC) + ADJQ (<) / C /
/ ADJQ (<) / I /

Ces règles partielles servirons pour guider les recherches de règles et pour les tests de cohérence des règles de résolution des homographies.

5.2) Fonctionnement:

Pour la suite nous avons séparé les homographies en deux types : les homographies simples (le voisinage n'est pas homographe) et homographies multiples (le voisinage contient des homographies) car les traitements de ces deux types d'homographies reposent sur des principes différents. Par voisinage, nous entendons le voisinage immédiat, c'est-à-dire les mots adjacents. Dans les deux cas nous ne traitons le texte que quand c'est nécessaire, c'est-à-dire quand la morphologie a repéré une homographie. Les principes des traitements sont identiques aux regroupements : nous disposons à tout instant d'une fenêtre sur le texte et nous recherchons des règles qui coincident avec cette fenêtre. L'expansion de la fenêtre est guidée par les règles partielles.

5.2.1) Homographies simples

Nous pouvons diviser les règles en deux groupes correspondant à des traitements différents :

- Les règles d'interdiction et d'implication
- Les règles de choix contextuel

Théoriquement les traitements pourraient être faits en parallèle car nous pouvons assurer la cohérence interne de l'ensemble des règles

de résolution des homographies. Dans cette réalisation nous les exécutons dans l'ordre précité. D'autre part pour des raisons d'efficacité nous n'effectuerons le deuxième traitement qu'en cas d'échec du premier.

5.2.1.1) Interdiction et implication

Cette phase a pour but d'éliminer certaines CL parmi les CL de l'homographie. Si, en fin de traitement, il ne reste qu'une CL non interdite l'homographie sera résolue sinon il faudra continuer le traitement.

Le fonctionnement s'apparente à celui des regroupements, nous avons toujours les deux phases :

- Recherche de l'ensemble des règles applicables
- Application des règles

Recherche des règles applicables

La recherche de l'ensemble des règles applicables dans la base de données des règles est guidée par le texte. Dans le cas d'homographies simples, nous essayons d'identifier une suite de CL présente dans le texte (image du texte). Nous disposons à tout moment d'une fenêtre sur le texte, l'homographie se trouve dans cette fenêtre qui représente une configuration courante. L'expansion de cette fenêtre est fonction de la présence de règles partielles dont la partie gauche coincide avec elle et du sens dans lequel chacune des règles partielles demande l'expansion. Nous procédons comme suit :

- Si la suite courante des CL apparaît dans le dictionnaire des règles en tant que règle partielle, il peut exister des règles applicables dans la configuration actuelle. Il faut donc continuer les essais d'identification.

- Si la suite courante des CL apparaît comme une règle globale, cette règle peut s'appliquer après vérification du contexte (variables).

- Si la suite courante n'apparaît pas dans le dictionnaire des règles, il est inutile d'essayer de trouver une règle applicable, on arrête donc le traitement.

Pour l'écriture de cet algorithme nous supposons que nous disposons des mêmes primitives que pour l'algorithme 1, paragraphe 4.2, c'est à dire :

- MEMORISE : même signification.

- MOT_SUIVANT : même signification.

- VOIR_A_GAUCHE : identique au "PARTIEL" de l'algorithme 1, mais ne concerne que les règles partielles ayant l'indicateur "<" (voir à gauche).

- VOIR_A_DROITE : identique au précédent avec l'indicateur ">".

- MOT_PRECEDENT (FENETRE) : symétrique de MOT_SUIVANT, avec le mot précédant dans le texte la fenêtre passée en paramètre.

- IDENTIFIE (FENETRE) : similaire à l'algorithme 1, mais positionne VOIR_A_GAUCHE, VOIR_A_DROITE et GLOBAL.

- HOMOGRAPHIE (MOT) : fonction délivrant le résultat "VRAI" si le mot passé en paramètre est une homographie, "FAUX" sinon.

- BLOQUE : logique mémorisant si l'expansion de la fenêtre a été bloquée par la présence d'une autre homographie. En cas de blocage, le traitement sera celui d'une homographie multiple (cf paragraphe 5.2.2).

- RETOUR : primitive rendant le contrôle à la procédure appelante.

Algorithme 2

```
PROCEDURE HOMOGS ( FENETRE, MOT_COURANT );
VARIABLE : CL_COURANTE ;
           MOT_SUIV ;

DEBUT

     IDENTIFIE ( FENETRE );

     SI GLOBAL ALORS MEMORISER ;

     SI VOIR_A_DROITE ALORS
     DEBUT

          MOT_SUIV := MOT_SUIVANT ( FENETRE );

          SI HOMOGRAPHIE ( MOT_SUIV ) ALORS
          DEBUT
               BLOQUE := VRAI ;
               RETOUR ;
          FIN ;

          CL_COURANTE := PREMIER ( MOT_SUIV);
          HOMOGS( FENETRE !! CL_COURANTE);

     FIN;

     SI VOIR_A_GAUCHE ALORS
     DEBUT

          MOT_SUIV := MOT_PRECEDENT ( FENETRE );

          SI HOMOGRAPHIE ( MOT_SUIV ) ALORS
          DEBUT
               BLOQUE := VRAI ;
               RETOUR ;
          FIN ;

          CL_COURANTE := PREMIER ( MOT_SUIV);
          HOMOGS( CL_COURANTE !! FENETRE );

     FIN;

FIN;
```

Cette procédure est lancée pour chacune des CL de l'homographie à résoudre. Par exemple, pour résoudre une homographie sur "bien" (substantif : "SUBC" ou adverbe : "ADV") nous lancerons :

```
HOMOGS ( "ADV", MOT_COURANT );
HOMOGS ( "SUBC", MOT_COURANT ) ;
```

Nous disposerons donc à la fin de cette étape de recherche de l'ensemble de toutes les règles applicables dans la configuration courante.

Exemple:

Soit le texte : " je marche vite "

La morphologie donne :

```
je      :   CL = PROS, VAR = SIN
marche  :   CL = VERB, VAR = SIN   et    CL = SUBC, VAR = SIN,FEM
vite    :   CL = ADV , VAR = ...
```

Les règles globales du dictionnaire sont :

```
/PROS ( NBR ) + VERB (NBR) / I / : accord en nombre entre le pronom
                                   personnel sujet et le verbe.
/PROS + SUBC / I /               : Un substantif ne peut suivre un
                                   pronom personnel sujet.
/ADJQ(NBR) + SUBC(NBR) / I /     : Accord en nombre entre un adjectif
                                   et un substantif.
```

Les règles partielles seront donc :

```
/PROS (>) / I /
/ADJQ (>) / I /
/VERB (<) / I /
/SUBC (<) / I /
```

La séquence des opérations sera :

1) chaîne courante = VERB on trouve : / VERB (<) / I /
   " = PROS+VERB " :/PROS(NBR)+VERB(NBR)/I/
   (règle globale à mémoriser)
   Fin d'identification : plus de règle partielle.

2) chaîne courante = SUBC on trouve : / SUBC (<) / I /
   " " = PROS + SUBC " : / PROS + SUBC /I/
   (Règle globale à mémoriser)

Fin d'identification : plus de règle partielle.

Nous disposerons pour cet exemple de deux règles applicables :

/PROS(NBR)+VERB(NBR)/I/
/ PROS + SUBC / I /

Application des règles :

Cette étape assure le contrôle des variables pour les règles conditionnelles et l'application de toutes les règles en effectuant un marquage des CL concernées. Ce marquage différencie les CL interdites et impliquées de façon à permettre par la même occasion un contrôle du texte.

A la suite de ce marquage nous voyons si l'homographie est résolue. Comme nous supposons que le texte est correct, c'est à dire qu'il existe toujours une solution au problème du choix de la "bonne" CL, l'homographie est résolue si :

- Une seule CL est non interdite.
- Une CL est impliquée.

Exemple:

Dans l'exemple précédent, les deux règles :

1) / PROS + SUBC / I /
2) / PROS(NBR) + SUBC(NBR) / I /

s'appliquent :

la règle 1) interdit la CL SUBC
la règle 2) n'a aucun effet , l'accord en nombre étant vérifié.

Il ne reste qu'une seule CL possible : VERB, l'homographie est donc résolue.

Remarque:

Nous pourrions appliquer immédiatement chaque règle et arrêter le traitement dès que l'homographie serait résolue. Mais cette

mémorisation de toutes les règles applicables nous permet d'effectuer un contrôle du texte et des règles à moindre frais. Les actions générées par les règles sont très simples et le traitement des homographies simples est très rapide; ce contrôle ne pénalise donc pas trop le temps d'exécution. Par contre si nous décelons une incohérence (toutes les CL interdites), soit le texte, soit une des règles est erroné. Nous pouvons donc en avertir l'utilisateur et lui permettre de corriger les erreurs en cours de traitement.

Pour la même raison, les résultats du traitement de cette première partie sont conservés pour la seconde partie afin de permettre un nouveau contrôle de saisie : une CL choisie ne peut en aucun cas avoir été interdite lors du premier traitement.

5.2.1.2) Choix contextuel

Contrairement au traitement des interdictions et des implications où nous considérions successivement toutes les CL homographes, ici nous allons étudier l'homographie dans son ensemble. L'identification sera lancée non plus sur chacune des CL mais sur l'homographie elle-même considérée comme un ensemble de CL. Le mécanisme reste le même, ce sont toujours les règles partielles, et donc le texte, qui guident les recherches.

La phase d'identification donne toujours un ensemble de règles applicables afin de permettre dans le même temps un contrôle de saisie. L'algorithme de recherche des règles est identique à l'algorithme 2, seul le lancement initial de la procédure "HOMOGS" diffère : au lieu de lancer cette procédure avec chacune des CL homographes, nous la lançons avec l'ensemble de ces CL. Par exemple pour une homographie sur la chaîne "été", ambiguë entre un substantif ("SUBC") et le participe passé de l'auxiliaire "être" ("PXET"), l'appel initial sera :

```
HOMOGS ( "(SUBC, PXET)", MOT_COURANT );
```

Chaque règle de choix spécifie en partie droite la CL à choisir dans la configuration actuelle. Le contrôle de cohérence vérifie qu'une CL choisie n'est pas interdite et qu'une CL impliquée n'est pas écartée.

Exemple:

Soient la chaîne : "J'ai été trouvé" et la règle de choix :

/PROS + XAV + (SUBC , PXET) + PPAS / C / PXET /

"j'" : CL = PROS : pronom personnel sujet
"ai" : CL = XAV : auxiliaire avoir
"été" : CL = SUBC : substantif
CL = PXET : participe passé de l'auxiliaire être.
"trouvé" : CL = PPAS : participe passé.

Le traitement est :

chaîne recherchée	règle trouvée
(SUBC , PXET)	/(SUBC,PXET)(<)(>)/ C /
(SUBC , PXET)+PPAS	/(SUBC,PXET)+PPAS(<)/C/
XAV+(SUBC,PXET)+PPAS	/XAV+(SUBC,PXET)+PPAS(<)/C/
PROS+XAV+(SUBC,PXET)+PPAS	/PROS+XAV+(SUBC,PXET)+PPAS/C/PXET/ (règle globale)

Au vu de la partie droite de cette règle, nous choisissons pour le mot "été" la CL : PXET (participe passé de l'auxiliaire "être").

5.2.2) Homographies multiples

Les cas où plusieurs homographies se succèdent dans le texte nous posent plusieurs problèmes liés à la "philosophie" du système : toutes les analyses portent sur le voisinage de l'homographie dans le texte, si ce voisinage est figé les traitements sont assez simples, mais, s'il est lui-même homographe, deux possibilités se présentent :

- Si il existe une règle de choix prenant en compte toutes les homographies de cette séquence, nous l'appliquerons et traiterons ainsi globalement ce problème.

- Si il n'existe pas de telle règle, il faut énumérer toutes les solutions possibles pour se ramener à une configuration utilisable et vérifier l'unicité d'une solution éventuelle.

Ces deux types de traitement s'apparentent aux mécanismes utilisés précédemment pour les homographies simples :

- traitement global (règles de choix contextuel).
- traitement partiel (règles d'interdiction et d'implication).

Le traitement partiel est un traitement combinatoire où nous envisageons toutes les possibilités de successions de CL et où nous déterminons si ces successions sont interdites. Le traitement global permet de rester dans un mécanisme très simple de reconnaissance de configuration et de choix de certaines CL pour chacune des homographies présentes dans cette configuration.

Pour des raisons bien évidentes de coût nous allons favoriser le traitement global par rapport au traitement partiel, nous allons donc en priorité rechercher des règles de choix applicables. Ces règles de choix peuvent ne pas affecter toutes les homographies se suivant dans le texte, nous serons donc amenés à faire un traitement partiel pour les homographies non concernées.

L'algorithme sera donc un algorithme combinatoire qui consistera, pour chacune des homographies rencontrées, à rechercher une règle de choix applicable, règle qui la traitera d'une façon globale. S'il n'en existe pas, ou si toutes les homographies ne sont pas résolues, il utilisera les différentes CL homographes dans un traitement partiel.

Pour l'écriture de cet algorithme, nous supposons disposer des primitives et des variables globales que nous avons définies lors de l'écriture des algorithmes 1 et 2 ainsi que de :

- TOUT_RESOLU : fonction délivrant les résultats "VRAI" si toutes les homographies concernées sont résolues, "FAUX" sinon.

- SENS : paramètre pouvant prendre les valeurs "GAUCHE" et "DROITE", indiquant le sens dans lequel doit s'étendre la fenêtre.

- ENS_CL (MOT) : fonction délivrant sous la forme d'une chaîne l'ensemble des CL du mot homographe passé en paramètre.

- APPLIQUE_REGLE : procédure appliquant les règles globales repérées par "IDENTIFIE". Cette application consiste à éliminer des possibilités de successions de CL.

Algorithme 3

```
PROCEDURE HOMOGM ( FENETRE, MOT_COURANT, SENS ) ;
VARIABLES : CL_COURANTE ;
            MOT_SUIV ;
            FENETRE_COURANTE ;
DEBUT

SI HOMOGRAPHIE ( MOT_COURANT ) ALORS
DEBUT
              (* ESSAI TRAITEMENT GLOBAL *)

    SI SENS = GAUCHE ALORS
      FENETRE_COURANTE := ENS_CL ( MOT_COURANT ) !! FENETRE
    SINON
      FENETRE_COURANTE := FENETRE !! ENS_CL ( MOT_COURANT ) ;

    IDENTIFIE ( FENETRE_COURANTE );

    SI GLOBAL ALORS
    DEBUT
       APPLIQUE_REGLE;
       SI TOUT_RESOLU ALORS RETOUR ;
    FIN;

    SI VOIR_A_GAUCHE ALORS
    DEBUT
       HOMOGM(FENETRE_COURANTE,MOT_PRECEDENT(FENETRE_COURANTE),GAUCHE);
       SI TOUT_RESOLU ALORS RETOUR;
    FIN;

    SI VOIR_A_DROITE ALORS
    DEBUT
       HOMOGM(FENETRE_COURANTE,MOT_SUIVANT(FENETRE_COURANTE),DROITE);
       SI TOUT_RESOLU ALORS RETOUR;
    FIN;

FIN;

          (* ECHEC TRAITEMENT GLOBAL, TRAITEMENT PARTIEL *)

CL_COURANTE := PREMIER ( MOT_COURANT );

TANTQUE CL_COURANTE <> NIL FAIRE
DEBUT

    SI SENS = GAUCHE ALORS
       FENETRE_COURANTE := CL_COURANTE !! FENETRE
    SINON
       FENETRE_COURANTE := FENETRE !! CL_COURANTE ;

    IDENTIFIE ( FENETRE_COURANTE );
```

```
        SI GLOBAL ALORS
        DEBUT
            APPLIQUE_REGLE;
            SI TOUT_RESOLU ALORS RETOUR ;
        FIN;

        SI VOIR_A_GAUCHE ALORS
        DEBUT
           HOMOGM(FENETRE_COURANTE,MOT_PRECEDENT(FENETRE_COURANTE),GAUCHE);
           SI TOUT_RESOLU ALORS RETOUR;
        FIN;

        SI VOIR_A_DROITE ALORS
        DEBUT
           HOMOGM(FENETRE_COURANTE,MOT_SUIVANT(FENETRE_COURANTE),DROITE);
           SI TOUT_RESOLU ALORS RETOUR;
        FIN;

        CL_COURANTE := SUIVANT ( MOT_COURANT, CL_COURANTE );

    FIN;

    FIN  (* HOMOGM *) ;
```

Si le programme ne parvient pas à résoudre toutes les homographies, il interroge l'utilisateur et lui demande de résoudre la première homographie non résolue "utile" (au sens de la génération de mots clés). Nous relançons alors le traitement pour les homographies suivantes.

Exemple

Soit la phrase: "je parle bien français."

La morphologie donne :

"Je" : CL = PROS , VAR = SIN
"parle" : CL = VERB , VAR = SIN
"bien" : CL=SUBC, VAR=MAS,SIN et CL=ADV
"français" : CL=SUBC, VAR=MAS,SIN,PLU et CL=ADJQ,VAR=MAS,SIN,PLU

Nous allons voir, dans cet exemple, la résolution du groupe de deux homographies "bien" et "français" suivant la présence ou l'absence de la règle globale :

/VERB+(SUBC,ADV)+(SUBC,ADJQ)/C/ADV+ADJQ/

Nous disposerons de toute façon des règles :

1) /SUBC + SUBC/ I /
2) /VERB + ADV + (SUBC , ADJQ) / C / ADJQ /
3) /VERB + (ADV , SUBC) + ADJQ / C / ADV /

Les quatres possibilités à étudier sont:

A) ADV + ADJQ
B) ADV + SUBC
C) SUBC + ADJQ
D) SUBC + SUBC

Dans les deux cas, les traitements seront :

chaîne recherchée	règle trouvée
(ADV , SUBC)	(ADV , SUBC)(<)(>)
VERB + (ADV,SUBC)	VERB+(ADV,SUBC)(>)
VERB + (ADV,SUBC)+(SUBC,ADJQ)	VERB+(ADV,SUBC)+(SUBC,ADJQ) / C /...
	(Si cette règle est présente le traitement est fini avec le choix de la possibilité A)

Si cette règle est absente, nous ne trouvons rien, donc le traitement se poursuit comme suit :

VERB + (ADV,SUBC)+(SUBC,ADJQ)	Rien
VERB + (ADV,SUBC) + SUBC	Rien
VERB + (ADV,SUBC) + ADJQ	/ VERB +(ADV,SUBC)+ADJQ / C / ADV /
	(Ce choix nous permet d'interdire la possibilité C et d'autoriser A)
ADV	ADV (<)(>)
VERB + ADV	VERB + ADV (>)
VERB + ADV +(SUBC,ADJQ)	/ VERB + ADV +(SUBC,ADJQ)/ C / ADJQ /
	(Ce choix nous permet d'interdire la possibilité B et d'autoriser A)
VERB + ADV + SUBC	Rien
VERB + ADV + ADJQ	Rien
SUBC	SUBC (<)(>)
SUBC + (SUBC,ADJQ)	Rien

SUBC + SUBC / SUBC + SUBC / I /

(Cette règle nous permet d'interdire la possibilité D, la seule solution possible est donc A, le traitement est fini)

Dans les deux cas nous arrivons au même résultat, si la règle globale est présente nous y arrivons immédiatement, sinon nous multiplions les accès dictionnaire. Nous trouvons donc :

"Je" : CL = PROS , VAR = SIN
"parle" : CL = VERB , VAR = SIN
"bien" : CL = ADV
"français" : CL = ADJQ , VAR = MAS,SIN,PLU

Remarques

Dans le cas où plusieurs homographies se suivent dans le texte, nous ne pouvons en résoudre une indépendamment des autres(sauf éventuellement les extrêmes), ces traitements peuvent donc nous amener à essayer de résoudre des homographies qui a priori n'étaient pas utiles pour l'application documentaire (aucune génération de mots-clés, génération identique, etc...). Mais il faut cependant remarquer que nous n'essayons de les résoudre que si elles sont nécessaires à des résolutions utiles.

Ces problèmes d'homographies "liées", c'est à dire dont la résolution doit se faire en bloc, se présentent soit quand plusieurs homographies se suivent dans le texte et que nous ne pouvons résoudre directement ce groupe d'homographies (soit par une règle globale, soit en résolvant les extrêmes indépendamment du reste et en itérant), soit quand la résolution d'une homographie est bloquée par la présence d'une autre homographie pas nécessairement adjacente. Le repérage des homographies "liées" non-adjacentes est effectué à l'aide de la variable "BLOQUE" de l'algorithme 2.

Nous essayons toujours dans cette résolution d'homographies multiples de nous ramener en priorité à des cas plus simples de résolution d'homographies simples. Pour cela, quand nous avons le choix, nous favorisons certains sens d'expansion : à gauche pour la première homographie du groupe et à droite pour la dernière.

Le traitement partiel, fortement combinatoire et donc d'un coût très élevé, devrait inciter l'utilisateur à utiliser au maximum le traitement global et pour cela à écrire au maximun des règles de choix contextuel.

Cet algorithme est applicable pour la résolution des homographies "simples". Nous avons conservé ces deux types de traitements car l'algorithme 2, en recherchant toutes les règles applicables, permet de réaliser un contrôle du texte. C'est pour cette raison que nous n'avons pas uniformisé les traitements.

6) COMPILATEUR DE REGLES PRESYNTAXIQUES

6.1) Fonctions

Ce compilateur assure la gestion de la base de données des règles de la présyntaxe, ces règles sont stockées dans un dictionnaire PIAF ((GRA1), (GRA4)). Le noyau de ce module est celui de la gestion du dictionnaire morphologique (ajout et destruction de mots, listage, accès aux informations, ...) , ce qui nous a permis d'uniformiser les traitements et les structures de données manipulées. Les temps d'accès aux informations sont toujours très adaptés aux traitements.

Les nouveautés et changements par rapport aux modules de gestion du dictionnaire morphologique sont :

- Analyse lexicale et syntaxique des règles données en symbolique par l'utilisateur afin d'assurer un contrôle de saisie.
- Traduction des règles en code interne.
- Génération des règles partielles.
- Vérification de la cohérence de l'ensemble des données.

Les analyses et traductions ne posent aucun problème, nous ne les développerons donc pas ici. Le code interne est un code où chaque CL est connue sous son numéro interne et les variables sous la forme de masques de bits.

La génération des règles partielles a été décrite au paragraphe 3, nous n'y reviendrons donc pas.

Le dernier point est sans aucun doute le plus important. L'ensemble des données est, par définition, évolutif en raison, d'une part, de l'évolution permanente du dictionnaire morphologique qui, en tant que dictionnaire d'une langue naturelle, n'est jamais figé, et, d'autre part, d'une évolution propre, de nouvelles règles étant introduites à chaque nouvelle homographie rencontrée.

6.2) Cohérence

Les différents tests de cohérence effectués sont décrits dans le chapitre 2 (paragraphe A.4.7).

Lors de chaque ajout de règle, nous vérifions que la règle à inclure est cohérente avec les autres règles. Pour cela il faut vérifier que cette règle ne contient pas de règle contradictoire et qu'elle n'est elle-même pas incluse dans une règle contradictoire.

Pour le premier point, il suffit de considérer la chaîne des CL de la partie gauche comme l'image d'un "texte" et de lancer la recherche des règles applicables sur ce "texte". Une fois obtenu l'ensemble des règles applicables, les tests sont immédiats.

Pour le deuxième point, l'ensemble des règles partielles nous indique si la partie gauche à insérer apparaît dans une règle de la base par une simple identification dans le dictionnaire. Si c'est le cas et pour obtenir un ensemble de règles contenant cette partie gauche, il suffit de compléter la partie gauche dans le sens indiqué par les partielles avec toutes les possibilités de CL (elles sont en nombre très restreint : une vingtaine), de vérifier la cohérence des deux règles (tests immédiats) et de relancer le traitement.

6.3) Stockage des informations

Le dictionnaire PIAF est destiné à mémoriser des chaînes de caractères. Nous ne le décrirons pas en détail ici, mais nous rappellerons d'une façon générale son fonctionnement.

Les mots sont regroupés en classes. Ces classes sont construites automatiquement et leur nombre est un paramètre du programme. L'accès à ces classes s'effectue au moyen d'un arbre binaire (recherche dichotomique). Chacune des classes est organisée en listes chaînées "frère-fils". La liste des fils est une liste de mots dont la chaîne de caractères est incluse dans la chaîne de caractères du père (en commençant par la première lettre), ce qui permet de ne pas stocker cette chaîne. La liste des frères est la liste des mots de la classe ayant des chaînes de caractères différentes. Ces mots sont chainés par ordre alphabétique. La recherche dans ces listes s'effectue donc de manière séquentielle avec priorité à la liste des frères.

Il existe une possibilité de chainage circulaire entre des mots de ce dictionnaire.

Pour utiliser cette structure nous stockons les informations sous la forme :

- La chaîne des CL apparaissant en partie gauche.
- Un masque des variables, pour des règles les utilisant.
- La chaîne des CL de la partie droite si nécessaire.

Le chainage circulaire nous permet de relier ces trois chaînes. Les identifications s'effectuent donc en deux étapes :

- Identification de la chaîne des CL de la partie gauche.
- Parcours du chainage circulaire pour éventuellement trouver un masque des variables et une partie droite.

Les codages des règles sont de la forme :

- CL : un caractère contenant son numéro interne.
- VARIABLES : un masque de bit par apparition de spécification de variables dans la partie gauche.
- Partie droite : chaque CL apparaît sous son numéro interne, et cette chaîne est précédée d'un code spécial signifiant "partie droite de règle".

7) CONTROLE DE SAISIE

Le contrôle de validité du texte d'entrée est effectué en deux occasions. Dans un premier temps le module d'analyse morphologique vérifie que les mots rencontrés existent et sont bien construits par référence aux paramètres morphologiques. Ces tests portent sur l'existence des mots mais ne vérifient aucune propriété grammaticale. Ils permettent de repérer un grand nombre de fautes de frappe qui constituent la majorité des erreurs. Dans l'absolu ces tests peuvent paraitre suffisants, mais, comme nous avions le moyen de le faire à un très faible coût, nous avons créé un deuxième contrôle de validité

obtenu en recherchant pour chaque homographie simple rencontrée l'ensemble des règles applicables. L'ensemble de règles étant cohérent (vérification lors de l'ajout de chaque nouvelle règle), si nous aboutissons à une incohérence ou à une impossibilité, c'est que le texte ou une règle est erroné. Les incohérences décelées sont :

- Toutes les CL sont interdites.
- Une CL est interdite et impliquée.
- Choix d'une CL interdite.

Lorsqu'il repère une de ces configurations, le logiciel avertit l'utilisateur qui a la possibilité de corriger le texte ou de modifier la base de données des règles et de relancer le traitement.

Remarque :

Nous pourrions faire un contrôle plus strict en appliquant pour chaque homographie toutes les règles applicables (interdiction, implication et choix). Mais ce contrôle nous a paru excessivement cher par rapport au résultat espéré : les fautes grammaticales sont assez rares et nous les repérons généralement avec la morphologie. Celles que nous ne pouvons repérer sont pour la plupart dues à des fautes de frappe non décelables (un mot valide se transforme en un autre mot valide). Ces fautes de frappe non repérables jouent en général un rôle très minime pour le rappel des documents dans la base documentaire (production du même mot-clé). Nous avons donc réalisé le deuxième contrôle uniquement quand nous pouvions le faire à un coût réduit : c'est à dire lors de la résolution d'homographies simples.

CHAPITRE 4 : EXPERIMENTATION

Nous allons ici décrire les résultats, les remarques et les conclusions que nous a révélés une expérimentation du système sur quelques textes. Cette expérimentation a été effectuée par nos soins et tout ce qui suit devra être validé et évalué plus précisément par une expérimentation beaucoup plus large que celle que nous avons effectuée.

Nous avons voulu montrer que notre approche du problème des homographies pouvait donner de bons résultats, aussi toutes les mesures ne tiennent pas compte des propriétés caractéristiques de l'application documentaire, c'est à dire que nous ne faisons pas intervenir ici la génération des mots-clés : nous avons essayé de résoudre toutes les homographies repérées par la morphologie.

Les paramètres morphologiques utilisés sont ceux définis dans (COUD), ils correspondent à une analyse morphologique de textes français écrits en typographie riche (avec les voyelles accentuées). Les seules modifications que nous avons apportées à cette liste, sont la définition de CL particulières pour les mots "introductifs", c'est à dire :

IGN : Introductif de groupe nominal ("au", "à une", "aux", "du").
IGI : Introductif de groupe infinitif ("de le", "de les", ...)
IGIN: Introductif de groupe infinitif ou nominal ("à la", "de la", ...)

1) REGLES

Les performances du système dépendent des règles disponibles. Nous allons d'abord décrire les règles dont nous disposions pour cette expérimentation.

Nous avons beaucoup regretté de ne pas pouvoir collaborer avec des linguistes pour la définition de ces règles, aussi cette liste et les chiffres cités dans ce chapitre sont sujets à de nombreuses modifications.

1.1) Regroupements

Nous l'avons dit dans le chapitre 2, les regroupements ne sont qu'un traitement très simple qui nous permet d'accélerer la résolution des homographies. Les règles que nous avons définies sont donc des règles très simples qui ont pour buts principaux de :

- résoudre certaines homographies évidentes ne demandant qu'une analyse très restreinte du voisinage.

- réduire, dans la mesure du possible, le nombre de mots de la phrase (en vue d'une utilisation du logiciel comme prétraitement d'une analyse syntaxique complète).

Ces deux objectifs donnent trois grands types de règles :

1) Essai d'élimination des adverbes, ou réécriture d'un groupe de deux adverbes en un seul adverbe.

Exemple

```
/ ADV + ADV / R / ADV /        prise en compte du "très" : "très vite",
                                 "plus rapidement", ...
/ VERB(£) + ADV / R / VERB /   adverbe qualifiant le verbe :
                                 "il marche vite", ...
/ ADV + ADJQ(£) / R / ADJQ /   formes superlatives : "très grand",
                                 "très petit", ...
```

2) Prise en compte de l'homographie sur le participe passé de l'auxiliaire "être" par ces regroupements.

Exemple

```
/ XAV + PXET(£) (S) / R / XET / "a été", "ai été", ...
```

Prise en compte de la négation et d'une homographie sur le verbe.

```
/ NE + VERB(£) + PAS (S) / R / VERB /   "ne marche pas", "n'élève pas",
 ...
 .... Plusieurs règles identiques pour les verbes et auxiliaires.
```

3) Diminution du nombre de mots dans le groupe verbal, dans le cas de verbes au passé composé.

```
/ XAV(£) + PPAS / R / VERB /            "a acheté", "a vendu", ...
/ XAV(£) + ADV + PPAS / R / VERB /      "a bien acheté", ...
```

1.2) Résolution des homographies

1.2.1) Règles d'accord

Ces règles sont simples à écrire : elles demandent un accord en genre et en nombre entre différents mots. Le principal problème dans notre cas est que le système ignore la structure de groupe syntaxique : il connaît simplement des juxtapositions de mots. Il faut donc que toutes les CL présentes dans la règle appartiennent au même groupe, sinon imposer un accord n'a aucune signification.

Exemple

Accord en genre et en nombre entre l'article et le substantif :

```
/ ARTD(GNR) + SUBC(GNR) / I / "le ferme" ==> "ferme" n'est pas
                                              un substantif
/ ARTD(NBR) + SUBC(NBR) / I / "les élève" ==> "élève" n'est pas
                                              un substantif
/ ARTD(GNR) + ADJQ(GNR) / I / "le ferme" ==> "ferme" n'est pas
                                              un adjectif
...
```

Remarque

Ces règles sont assez fastidieuses à écrire, car il faut examiner toutes les formes possibles où cet accord doit être vérifié. Leur efficacité est très intéressante surtout dans le cas d'homographies verbe-substantif, ou d'une façon plus générale dans le cas d'homographies mettant en jeu des CL ne pouvant pas appartenir au même groupe syntaxique.

1.2.2) Règles de choix

Contrairement aux règles précédentes, il est très difficile de définir ces règles a priori. En effet, elles dépendent du modèle morphologique et du domaine d'application : une homographie peut être une homographie de la langue ou une homographie générée par le modèle morphologique.

D'autre part, ces règles sont assez peu naturelles et durant notre expérimentation nous les avons définies lors du traitement en rencontrant des homographies que les règles générales ne pouvaient résoudre. Il faut aussi remarquer que ces règles sont moins "dangereuses" que les règles globales car elles ont un domaine d'application très restreint, donc leur écriture sur un cas particulier dépasse rarement ce cas particulier.

Exemple

Homographie entre un substantif et un adjectif ou un participe passé dans un groupe nominal :

/ ARTD + (PPAS, SUBC) + ADJQ / C / SUBC /
"L'entreprise familiale ..." choix du substantif

/ ARTD + (ADJQ, SUBC) + XET / C / SUBC /
"Le français est une langue ..." choix du substantif

/ ARTD + (PPAS, SUBC) + PNT / SUBC /
"... l'entreprise ." choix du substantif

Cette dernière règle fait apparaître la CL : PNT, signifiant ponctuation (";",".",":","!","?",... sans la virgule). Ces ponctuations terminent une "phrase", ou plus exactement "ferment" toutes les structures "ouvertes" par la phrase les précédant. Le groupe syntaxique qui se place juste avant elles doit être complet, ce qui nous permet d'imposer un substantif dans un groupe nominal, un infinitif dans un groupe infinitif, etc... Ces règles et ces propriétés des ponctuations sont d'ailleurs assez importantes dans les systèmes statistiques (FLU).

Nous disposons de règles semblables pour les autres types de groupes.

/ VERB + (ADV , SUBC) + ARTD / C / ADV / "je parle bien le français"
...

1.2.3) Règles d'interdiction

Nous revenons ici à des règles beaucoup plus générales, donc définissables a priori (et, à notre avis uniquement comme cela en raison de leur portée). Nous avons pris en compte dans ces règles les propriétés des mots introductifs de groupes syntaxiques en français.

Exemple

```
/ PROS + SUBC / I /        "je marche" ==> substantif interdit
/ PROS + ADJQ / I /        "je ferme"  ==> adjectif interdit
/ NE   + SUBC / I /        "ne marche" ==> substantif interdit
/ NE   + ADJQ / I /        "ne ferme"  ==> adjectif interdit
/ IGN  + VERB / I /        "aux élèves" ==> verbe interdit
/ IGNI + VERB / I /        "de la marche" ==> verbe interdit
 ...
```

1.2.4) Règles d'implication

Nous avons noté toutes les notions d'obligation sous la forme de règles d'interdiction ou sous forme de règles de choix, nous avons vu, en effet, dans les précédents chapitres que ces règles n'étaient qu'une simplification d'écriture. Notre expérimentation ne comporte donc pas de règles de ce type. Nous pouvons alors nous poser la question de l'utilité de ce type de règles. Ces règles sont difficiles à définir en raison de leur caractère prédicatif. Pour l'étude d'une langue telle que le français, il est en effet très difficile de définir de telles règles mais nous pensons que pour d'autres applications elles pourraient être utiles. Ces applications pourraient être l'analyse de sous-ensembles simplifiés de la langue ou l'analyse de langages dont le modèle est plus simple que le modèle que nous manipulons.

Ces règles semblaient, a priori utiles pour noter les propriétés des mots introductifs, mais ces groupes peuvent avoir en français de nombreuses formes, par exemple un groupe nominal introduit par "au" peut commencer par un substantif, un adjectif, un adverbe , ... ce qui rend impossible la formalisation par une règle d'implication.

Exemple

"aux" introduit un groupe nominal, mais les formes suivantes sont possibles :

```
"aux élèves"              "aux" + SUBC
"aux bons élèves"         "aux" + ADJQ
"aux très bons élèves"    "aux" + ADV
```

Il faut cependant remarquer que l'on peut écrire de telles règles pour le français, par exemple un verbe suit toujours la séquence formée d'un pronom personnel sujet, d'un pronom personnel accusatif et d'un pronom personnel datif. Dans le système nous formaliserions ceci par :

/ PROS + PACC + PDAT > VERB / 0 / "je le lui" ==> verbe

Mais nous avons "divisé" la notion de verbe en trois classes lexicales : "VERB", "XAV" (auxiliaire "avoir") et "XET" (auxiliaire "être"). Or la séquence "je le lui", si elle demande un verbe, peut être suivie soit de l'auxiliaire "avoir" ("je le lui ai donné"), soit d'un verbe selon notre classification ("je le lui donne"). Les regroupements n'effectuant aucun traitement quand une homographie est présente et quand la règle applicable ne contient pas l'indicateur (S), ces deux séquences peuvent apparaître.

2) EVALUATION ET MESURES

2.1) Evaluation

Pour être intéressants, ces mécanismes doivent avoir une exécution rapide. Nous avons vu dans le chapitre 2 (partie B) que les traitements étaient équivalents à des automates d'états finis, mais sur un vocabulaire de parties. Notre but était de réaliser un traitement linéaire, c'est à dire dont le temps d'exécution est proportionnel au nombre de mots de la phrase.

Nous essayons ici d'évaluer les algorithmes que nous mettons en oeuvre en nombre d'accès au dictionnaire des règles. Les temps d'accès sont pratiquement constants pour un nombre donné de règles, aussi, les performances du système dépendront directement de ce nombre d'accès.

Pour chaque essai de regroupement la longueur de la fenêtre, et donc le nombre maximum d'identifications, est limité par la longueur des règles disponibles. En considérant que la longueur des règles est bornée à 4 ou 5 mots, nous pouvons dire que, si n est le nombre de mots de la phrase, le nombre d'identifications propres aux regroupements est de l'ordre de 5n. En effet les homographies sont assez rares (moins de 10% des mots), et le nombre moyen de CL de chaque homographie est compris entre 2 et 3, donc le nombre d'identifications générées par ces homographies est négligeable.

La résolution des homographies simples doit être, elle aussi, linéaire. Le nombre d'accès est limité par le nombre de CL de chaque homographie et la longueur des règles disponibles. Une grossière évaluation consiste à dire qu'une borne vraisemblable de ce nombre est obtenue en considérant que :

- le nombre moyen de CL homographes est 3 (La plupart des homographies sont des homographies entre deux CL et les homographies entre trois CL sont déjà assez rares).

- la longueur maximale des règles applicables est 5 : l'écriture des règles est à la charge de l'utilisateur mais il nous semble difficile de concevoir une majorité de règles plus longues.

Nous pensons donc, d'après ces chiffres, que le nombre d'accès nécessaire à la résolution d'une homographie simple doit être voisin de 15; ce nombre est obtenu par la formule :

nombre de CL * longueur maximale des règles

Le seul traitement combinatoire est la résolution des homographies multiples par énumération des différentes possibilités de succession de CL. Là aussi, pour chacune des possibilités, le nombre d'accès est limité par la longueur des règles disponibles. On peut donner une approximation de ce nombre par la formule :

N = (nCL ** ng) * l

où

N est le nombre d'identifications.
nCL est le nombre moyen de CL de chaque homographie.
ng est le nombre moyen de mots du groupe à résoudre.
l est la longueur moyenne des règles.
** est l'opérateur "puissance".

En considérant que le nombre moyen d'homographies liées est compris entre deux et trois, nous pouvons supposer que le nombre d'accès nécessaire à la résolution d'un groupe est voisin de 100.

Nous complèterons cette grossière évaluation en supposant que 10% des mots rencontrés sont homographes et que 10% de ces homographies sont "liées" dans le texte (c'est à dire nécessitent une résolution combinatoire).

Si n est le nombre de mots de la phrase, le nombre d'accès nécessaire à la résolution des homographies doit être inférieur à 3*n. Les regroupements entrainent un nombre proportionnel à n. Les traitements doivent donc être linéaires.

Les mesures que nous avons effectuées confirment ces chiffres. Mais, avec l'ensemble de règles réduit de cette expérimentation, le nombre d'accès dictionnaire était plus faible, il était de l'ordre de 4n, pour les regroupements et la résolution des homographies (pour 2000 mots environs, 8500 accès).

2.2) Mesures

Nous n'avons pu mener de réelle campagne de mesure, mais testé sur un corpus d'environ 4000 mots le système a permis de résoudre environ 70% des homographies (143 homographies résolues sur 212 repérées). Ces textes se répartissaient également entre des textes politiques (chronologie de la Documentation Française de l'année 1974) et des textes généraux. Ces homographies sont les homographies repérées par la morphologie, ces résultats sont donc des résultats relatifs au modèle morphologique.

En fait ces résultats se décomposent en 80% de réussite pour les textes généraux et 60% pour des textes politiques. Pourquoi cette différence ?

- Les textes généraux sont des textes "bien" construits alors que les textes politiques du corpus étaient des dépêches politiques adoptant le style "télégraphique" (phrases nominales, ...).

- Certains mots homographes apparaissent avec une grande fréquence dans les textes politiques. C'était le cas dans ces tests avec le mot "président", ambigu entre un substantif et un verbe. Il apparaît généralement sous la forme :

  "Monsieur x, président du ..."

  La virgule présente devant le mot "président" est une coupure, le contexte gauche du mot est donc inutilisable.

  Le contexte droit ne nous permet pas de résoudre "syntaxiquement" l'homographie : la construction verbe suivi de "du" est parfaitemet valide ("je bois du vin"), de même que la construction "substantif suivi de du". Nous ne pouvons donc rien dire avec les moyens mis en oeuvre dans cette réalisation.

Il faut remarquer que le problème posé par le mot "président" est un cas particulier de cette chonologie politique. En effet nous avons analysé un texte général qui est une préface d'un ouvrage de syntaxe destiné aux enseignents de français des écoles primaires. Le mot "élève" y apparaît toutes les deux lignes, ce mot est homographe entre un substantif et un verbe. Le système a pourtant donné dans ce cas là

d'excellents résultats (le texte comportait 660 mots dont 23 homographies, nous en avons résolues 21). En effet "élève" est employé dans la plupart des cas comme un substantif, or en français un substantif est généralement introduit par un article. Nous retrouvons en majorité des formes du types :

"pour les élèves" or "pour les" introduit un groupe nominal ou infinitif.
"aux élèves" or "aux" introduit un groupe nominal.
"des éléves" or "des" introduit un groupe nominal.
...

3) REMARQUES

Les inconvénients du système sont de deux types :

- Une première classe de problèmes est liée à l'efficacité du logiciel. Les traitements sont, par définition, des traitements partiels, aussi l'utilisateur a à sa charge la résolution des homographies non-résolues. Il ne faut pas que ces homographies lui paraissent évidentes.

- Une deuxième classe de problèmes est liée à l'utilisation du système. Il faut définir des règles de résolution d'homographies. Cette tâche est difficile et de cette définition dépend l'efficacité, mais aussi la validité du système.

Nous avons vu que nous résolvions en moyenne 70% des homographies présentes dans les textes analysés. Ces résultats, bien que déjà intéressants, sont, à notre avis insuffisants pour l'application documentaire. Nous laissons encore passer certaines homographies qui paraissent évidentes aux utilisateurs, en raison d'un manque de caractérisation des contextes d'applicabilité des règles. Nous l'avons déjà dit, nous sommes actuellement incapable de résoudre l'homographie sur le mot "président" dans des phrases du type :

Monsieur x, président du ...

La résolution est pourtant évidente à un être humain. Il nous faudrait pouvoir résoudre ces homographies sinon, malgré les résultats du système, l'aide aux utilisateurs n'est pas évidente. Nous avons vu les possibiliés de résolution de ce type d'homographie, il faudrait donc :

- Soit compléter et en partie redéfinir le modèle morphologique pour nos traitements. Cette solution ne nous paraît pas être la meilleure en raison de la multiplication des modèles morphologiques qu'elle implique.

- Soit donner de nouvelles possibilités de définition des contextes d'applicabilité des règles et de nouvelles possibilités de caractérisation des mots ou des homographies rencontrés, par exemple l'introduction de l'attribut "chaîne de caractères" nous parait pratiquement indispensable.

  Nous pourrions alors, par exemple, définir une règle :

  "," + HOMOGRAPHIE ("président") + "du" ==> choix du substantif

  qui permettrait de résoudre cette homographie.

La deuxième classe de problèmes, nécessite de s'interroger sur la méthodologie de définition des règles.

Les règles d'interdiction, d'accord de variables et d'implication sont des constantes de la langue. En tant que telles, nous pensons qu'il faut les définir a priori et en dehors de toute application, de la même façon que la grammaire morphologique dans PIAF est indépendante de l'application.

Les règles de choix sont un ensemble de connaissances pragmatiques, elles ont une portée très réduite (surtout si on introduit l'attribut "chaîne de caractères"). La définition de ces règles peut se faire de façon incrémentale.

Par définition, le système PIAFPS effectue un traitement partiel. Il restera toujours des homographies qu'il sera incapable de résoudre avec un traitement équivalent à un traitement d'états finis. Le problème est de connaître la limite de ces traitements : lorsqu'on rencontre une homographie non résolue, pourquoi est-elle non résolue ?

- Parce que l'ensemble des règles est incomplet, il suffit alors de rajouter la ou les règles nécessaires.

- Parce qu'elle n'est pas de la compétence du système. Il ne faut alors rien changer dans les règles, cette homographie dans ce contexte génèrera toujours une question à l'utilisateur.

Mais comment reconnaître ces deux cas? Ne rien faire dans le premier n'est pas très important, seule l'efficacité du système est en cause. Rajouter des règles dans le deuxième est très grave, si effectivement l'homographie n'est pas résoluble par les traitements mis

en oeuvre, la résoudre est une erreur, pouvant entrainer d'autres erreurs lors de traitements ultérieurs.

Ce problème est très important et, pour l'instant, nous n'y voyons pas de solution parfaite. A notre avis, la seule solution acceptable est de recommander une extrême prudence aux utilisateurs lors de l'ajout de chaque règle et, éventuellement de reporter cet ajout sous la responsabilité d'une personne compétente.

Enfin, en conclusion de cette remarque, PIAFPS est, à notre avis, assez efficace en fonction des techniques mises en oeuvre. Nous devrions dans une version plus évoluée atteindre une efficacité moyenne d'environ 80 à 85% et il faudra sans doute s'en tenir là.

En résumé, nous pouvons dire que les principaux problèmes d'utilisation du logiciel sont liés à la définition des règles de résolution des homographies et que toute amélioration et tout développement passe par la création d'une aide lors de cette étape.

4) EVOLUTIONS POSSIBLES

Nous avons envisagé plusieurs solutions au problème de la définition des règles. Ces solutions sont principalement de deux types :

- Définition automatique de règles.
- Aide à la définition des règles, l'ajout d'une nouvelle règle restant sous la responsabilité de l'utilisateur.

Nous avons vu dans le premier chapitre (paragraphe 4.3) qu'un apprentissage classique était impossible.

La génération automatique de règles à partir d'un analyseur syntaxique est très difficile car cet analyseur devrait être génératif (et l'on sait les problèmes que cela pose).

Nous pouvons donc déjà dire que la première solution nous parait irréalisable, la définition et l'ajout de règles dans PIAFPS devra rester sous la responsabilité de l'utilisateur.

Par contre le deuxième point est envisageable, notamment avec le principe de l'apprentissage : une analyse statistique des successions de catégories dans des textes pourrait repérer les constructions absentes, ou très rares et proposer ces successions à l'utilisateur. Resteraient de son ressort le fait d'en tenir compte et, dans l'affirmative, la définition des règles, c'est à dire le choix de la formalisation de ces interdictions.

Il nous parait indispensable de fournir une aide pour valider l'ensemble des règles disponibles. Pour cela il faudra prévoir un fonctionnement en mode "contrôle", vérifiant si aucune des interdictions présentes dans les règles n'apparaît dans des textes supposés corrects. Ceci correspond à une généralisation des traitements que nous effectuons en cours de résolution des homographies, traitements vérifiant l'absence d'incohérence.

5) CONCLUSION

Actuellement le système PIAFPS est capable de résoudre en moyenne 70% des homographies présentes dans un texte libre. Ce chiffre varie en fait entre 60 et 80 % suivant les textes traités : les performances sont intéressantes pour des textes généraux et "bien" construits, elles sont nettement moins bonnes pour des textes précis et assez "mal" construits (style télégraphique, phrases nominales, ...). Ces performances sont à notre avis encore insuffisantes et doivent pouvoir être améliorées.

Certaines améliorations sont dès à présent envisageables, ce sont :

- La prise en compte de nouveaux attributs tels que la chaîne de caractères ambigue.

- Création d'un mode "contrôle" permettant de confirmer ou d'infirmer par une expérimentation la validité des règles présentes dans le système.

Une fois ces améliorations effectuées nous pensons pouvoir atteindre un taux de résolution d'environ 80% à 90% dans tous les cas de figures, et les traitements étant, par définition, partiels, il faudra sans doute en rester là.

CONCLUSION

Nous avons présenté dans cette thèse certains mécanismes pour une étude des langues naturelles. Notre but n'était pas de définir un nouveau modèle d'analyse, mais plutôt de répondre à certains problèmes liés aux analyseurs classiques.

Ceux-ci sont généralement organisés autour d'une syntaxe. Ce type d'analyse est actuellement le traitement de plus haut niveau adaptable à une analyse de textes libres. Il présente de nombreux inconvénients liés principalement à sa complexité : le modèle hors-contexte (donc non-linéaire) et la présence de nombreuses ambiguïtés impliquent une complexité incompatible avec toute une classe de problèmes nécessitant un temps de réponse rapide.

Les principes généraux de cette étude sont les suivants :

- Nous nous plaçons dans un environnement d'analyse formelle de textes libres.

- Nous désirons une sûreté de fonctionnement maximale.

- Nous voulons résoudre, par un traitement rapide, les problèmes liés à la combinatoire générés par les homographies. Ce traitement sera évidemment partiel, mais le contexte interactif ou une analyse plus élaborée, pourront résoudre les problèmes que le système est incapable de traiter.

Nous avons défini un traitement partiel et de surface permettant d'éliminer très simplement les incohérences dues à des mauvais choix lors de la résolution des homographies. Il est inspiré des techniques de "pattern matching" et des systèmes de productions bien connus en intelligence artificielle. Ce traitement permet de résoudre très simplement la plupart des homographies. Les principaux points de cette réalisation sont :

- Traitement local : nous analysons le texte uniquement quand c'est nécessaire par une immersion dans le voisinage des homographies. Nous nous opposons ici aux analyses grammaticales "classiques" qui nécessitent une analyse globale de la phrase pour résoudre tous les problèmes.

- Le but du système est d'éliminer les incohérences dues à des mauvais choix lors de la résolution des homographies.

- La prise en compte des homographies comme des objets du langage permet de traiter simplement certains points particuliers de la langue et notamment certaines conditions particulières liées à l'emploi de certains mots.

Nous avons généralisé ici les principes mis en oeuvre dans le logiciel PIAF, c'est à dire des traitements basés sur l'élimination des incohérences et non sur une interprétation des textes.

Nous avons cherché à prouver, dans cette étude, la faisabilité de ce type de traitement appliqué à l'analyse des langues naturelles. Cependant une utilisation effective du logiciel PIAFPS demande quelques développements :

1) Dans un premier temps il faudra introduire d'autres moyens de particulariser les contextes d'applicabilité de chaque règle. Cette particularisation passe par la création et la prise en compte de nouveaux attributs.

2) La définition des règles de résolution des homographies est actuellement entièrement à la charge de l'utilisateur; il est nécessaire de lui offrir des outils permettant, non pas une automatisation de cette phase, mais une aide pour vérifier la validité de l'ensemble des règles.

3) Le mécanisme des regroupements de portions de textes par une simple identification de successions de mots demande à être développé tant au plan pratique qu'au plan théorique. Dans cette réalisation ces regroupements n'ont pas une grande utilité, mais si cette analyse pré-syntaxique était associée à une analyse syntaxique complète, leur utilité serait évidente.

Dans l'application documentaire, les traitements présyntaxiques sont le module de plus haut niveau du système : tous les problèmes non résolus sont à la charge de l'utilisateur. Nous pensons pourtant que la principale utilité de ces mécanismes de filtrage et de regroupements pourrait être un pré-traitement à une analyse syntaxique complète. Cette analyse, déchargée de la construction des structures évidentes et de la résolution de la plupart des homographies, serait alors beaucoup plus rapide et utilisable dans des environnements que sa complexité lui interdit actuellement.

L'analyse pré-syntaxique que nous avons défini ici, n'est pas propre à l'application d'indexation automatique de documents. Ses domaines d'application sont, en fait, tous les problèmes demandant une analyse formelle de textes libres et ne nécessitant qu'une compréhension limitée. Parmi ces applications nous pouvons citer la recherche linguistique, l'analyse documentaire et donc la bureautique, et enfin la synthèse de la parole. Dans ce domaine, monsieur Gerard BAILLY réalise actuellement, dans le cadre d'une thèse préparée à l'ENSERG, une analyse de textes libres comprenant une analyse morphologique, une analyse pré-syntaxique et une analyse syntaxique. Nous attendons avec intérêt les conclusions de son étude.

En conclusion, les mécanismes de l'analyse présyntaxique que avons mis en oeuvre dans cette thèse restent une approche pragmatique du problème. Les connaissances sont une accumulation de faits, et le système assure la résolution de cas particuliers. La convergence est assurée par des interventions de l'utilisateur, mais qui, à part lui, peut prendre des décisions dans un domaine aussi complexe que l'analyse des langues naturelles ? D'ailleurs tous les traitements linguistiques ne sont-ils pas basés sur la mise en oeuvre de connaissances partielles et imparfaites ?

ANNEXE 1 : INTERFACE MORPHOLOGIE-PRESYNTAXE

Nous allons décrire rapidement les fonctions et les mécanismes de l'interface "Morphologie-présyntaxe" et donner la syntaxe exacte des règles de réécriture des informations fournies par la morphologie.

1) PRESENTATION

Nous l'avons vu au chapitre 3, paragraphe 4.1, l'universalité de la morphologie entraine plusieurs niveaux de description du comportement syntaxique des mots. Physiquement ces informations se trouvent :

- Dans la grammaire de la langue, sous forme de CL et de variables calculées à partir d'informations associées au modèle de construction des mots.

- Dans le dictionnaire de la langue, sous forme d'indicateurs associés à la racine d'un mot.

Les informations associées aux modèles sont des informations communes à la classe d'équivalence regroupant les mots ayant les mêmes constructions morphologiques. Les indicateurs sont propres à un mot. Cette interface est destinée à regrouper ces informations de façon à caractériser le comportement syntaxique d'un mot par une classe lexicale (CL) et par des variables.

Exemple:

Les auxiliaires "avoir" et "être", bien qu'étant des verbes, entrainent des constructions différentes des autres verbes. Ils sont pourtant classifiés sous la CL : "VERB". L'information "auxiliaire" est notée sous la forme d'une variable. Nous voulons créer une CL "XAV" (auxiliaire "avoir") et une CL "XET" (auxiliaire "être").

Certains adjectifs se placent avant, après ou indifféremment avant ou après le substantif qu'ils qualifient. Nous notons cette information sous la forme d'un indicateur "PS", pouvant prendre les valeurs "AV", "AP" et "" (chaîne vide). Nous voulons créer les CL "ADJA" (Adjectif se plaçant avant le substantif), "ADJP" (Adjectif se plaçant après le substantif) et "ADJQ" (Adjectif se plaçant indifféremment avant ou après le substantif).

Actuellement tous les pronoms personnels sont regroupés sous une même CL "POPL". Nous voulons différencier les pronoms personnels sujets ("je", "tu", ...) des autres pronoms personnels en créant une CL "PROS" (pronom personnel sujet).

La transformation se fait en fonction des CL, des variables et de différents indicateurs reconnus par la morphologie. Elle génère un nouveau couple (CL, variables).

Ce module assure en outre un filtrage des renseignements utiles à la présyntaxe : il conserve uniquement les informations utilisées par les regroupements et la résolution d'homographies.

Exemple:

Seules les variables GENRE (GNR) et NOMBRE (NBR) nous intéressent et pour la variable GNR seules les valeurs MASCULIN (MASC) et FEMININ (FEM) sont utilisées.

L'utilisateur donne un ensemble de règles de transformation (en symbolique) et un compilateur externe à PIAFPS génère les représentations internes des règles de transformation.

2) GRAMMAIRE DE TRANSFORMATION

La grammaire de transformation comporte trois parties :

- Déclarations des CL utilisées par la présyntaxe.
- Déclarations des variables utilisées par la présyntaxe.
- Déclarations des règles de transformation.

2.1) Déclaration des CL

Format:

<DECLARATION DE CL> ::= CL := <CL> , <suite de CL>.

<suite de CL> ::= <CL> , <suite de CL> ! <rien>
<CL> ::= nom d'une CL qui sera connue par la présyntaxe.

Toutes les CL utilisables dans la présyntaxe doivent être déclarées. Ces déclarations ont pour but de définir un code interne pour chacune des CL. Ce code sera celui de la morphologie pour les CL qui existaient déjà, et un nouveau code pour les CL créées par cette interface.

2.2) Déclaration des variables

Format

<DCL variables>::= <DCL noms de variables> . <DCL valeurs de variables>.

<DCL noms de variables> ::= VAR := <NOM> , <suite de NOM> .
<suite de NOM> ::= <NOM> , <suite de NOM> ! <NOM>
<NOM> ::= Nom d'une variable morphologique utilisée par la présyntaxe.

<DCL valeurs de variables> ::= <DCL VAL> . <Suite DCL VAL> .
<Suite DCL VAL> ::= <DCL VAL> . <Suite DCL VAL> . ! <rien>
<DCL VAL> ::= <NOM> := <VALEUR> , <Suite de VALEUR> .
<Suite de valeur> ::= <VALEUR> , <Suite de VALEUR> ! <rien>
<VALEUR> ::= Nom d'une valeur de variable morphologique utilisée par la présyntaxe.

Seules les variables déclarées seront utilisées par la présyntaxe et pour ces variables les valeurs utilisées seront :

- Si les valeurs de la variable sont redéclarées, on utilisera ces nouvelles valeurs.

- Si les valeurs ne sont pas redéclarées, on utilisera les valeurs qu'avait cette variable dans la morphologie.

2.3) Déclaration des règles de transformation

Format:

<REGLE> ::= / <CLMORP> [+VAR(<SVNV>)] [+<NIPS>(<VIPS>)] / <CLPS> /

<CLMORP> ::= CL MORPhologique.
<SVNV> ::= Suite de Valeurs ou de Noms de Variables.
<NIPS> ::= Nom d'Indicateur PréSyntaxique
(issu du dictionnaire morphologique).
<VIPS> ::= Valeur de l'Indicateur PréSyntaxique.
<CLPS> ::= CL PréSyntaxique
(Déclarée dans la grammaire de transformation).

Chaque règle spécifie une CL morphologique (CLMORP) à transformer en une CL présyntaxique (CLPS). Cette transformation s'effectue si le contexte précisé dans la partie gauche est présent. Ce contexte comprend :

- Certaines variables ou valeurs de variables. Si seul le nom de la variable est précisé, la transformation s'effectue si une quelconque des valeurs de cette variable est présente, sinon elle s'effectue si la valeur spécifiée est présente.

- Certains indicateurs présyntaxiques issus du dictionnaire morphologique. La transformation s'effectue si la valeur précisée pour cet indicateur est présente.

Exemple de grammaire :

```
BEGIN

                              * Declaration des CL, Cette liste contient
                                les noms "classiques" décrits dans
                                (COUD)*
CL:=VERB, NE, PAS, PXAV, PXET, XAV, XET, PROS, APRO, ADV, ADJQ, ADJA,
ADJP, SUBC, SUBP, ARTD, PPAS, PPRS, PREP, COCO, SADJ, PNT.

VAR:=GNR, NBR.                * Declaration des variables utilisées par
                                la Présyntaxe *
GNR:=MAS, FEM.                * Declaration des valeurs de variables :
                                NBR gardera les valeurs fournies
                                par la morphologie *
```

```
* Déclaration des règles de transformation *

/ADJQ + PS (AV) / ADJA /      * création de la CL "ADJA" quand
                                l'indicateur PS a la valeur AV   *
/ADJQ + PS ( AP) / ADJP /     * création de la CL "ADJP" quand
                                l'indicateur PS a la valeur AP   *
/VERB + VAR ( XAV) / XAV /    * création de la CL "XAV" quand un
                                verbe possède la variable "XAV   *
/VERB + VAR ( XET ) / XET /   * idem avec "XET" *
/PPAS + VAR ( XAV ) / PXAV /  * création de la CL "PXAV" : participe
                                passé de l'auxiliaire "avoir"    *
/PPAS + VAR ( XET ) / PXET /  * Idem avec "PXET" : participe passé de
                                l'auxiliaire "être"  *
/POPL + VAR ( SUJ ) / PROS /  * création de la CL "PROS" : pronom
                                personnel sujet *

END.
```

ANNEXE 2 : SYNTAXE DES REGLES PRESYNTAXIQUES

Le formalisme utilisé pour la description des règles est assez classique : les symboles entre "<" et ">" sont des non-terminaux de la description, le caractère "!" est un "ou" logique, les chaînes entre crochets ("[", "]") sont des parties optionnelles. Si il existe un risque d'ambiguité nous noterons entre guillemets (") les symboles devant apparaître dans la règle.

1) REGROUPEMENT

format général d'une règle:

<REGLE> ::= / <PG> / <TYPE> / <PD> /

<TYPE> ::= R
<PG> ::= <CL> [(#)] + <suite de CL> [(S)]
<suite de CL> ::= <CL>[(#)] + <suite de CL> ! <CL>[(#)]
<PD> ::= <CL>
<CL> ::= Nom d'une CL utilisée en présyntaxe.

2) RESOLUTION DES HOMOGRAPHIES

2.1)Règles d'interdiction et d'accord de variables

Format:

<REGLE> ::= / <PG> / <TYPE> /

<TYPE> ::= I
<PG> ::= <CL> [(<VAR>)] + <suite de CL>
<suite de CL> ::= <CL>[(<VAR>)] + <suite de CL> ! <CL>[(<VAR>)]
<VAR> ::= Nom ou valeur d'une variable utilisée en présyntaxe.
<CL> ::= Nom d'une CL utilisée en présyntaxe.

2.2) Règles d'implication

Format:

```
<REGLE>        ::= / <PG> / <TYPE> /

<TYPE>         ::= O
<PG>           ::= <PGD> ! <PGN>
<PGD>          ::= ">" <CL> + <CL>[(<VAR>)] + <suite de CL>
<PGN>          ::= <CL> [(<VAR>)] + <suite de CL> ">" <CL> + <suite de CL>
<suite de CL>  ::= <CL>[(<VAR>)] + <suite de CL> ! <rien>
<CL>           ::= Nom d'une CL utilisée en présyntaxe.
<VAR>          ::= Nom ou valeur d'une variable utilisée en présyntaxe.
```

La CL impliquée suit le symbole ">" de la partie gauche de la règle.

2.3) Règles de choix contextuel

Format:

```
<REGLE>          ::= / <PG> / <TYPE> / <PD> /

<TYPE>           ::= C
<PG>             ::= <MOT> + <suite de mots>
<suite de mots>  ::= <MOT> + <suite de mots> ! <MOT>
<MOT>            ::= <CL> ! <HOMOGRAPHIE>
<HOMOGRAPHIE>    ::= "[" <CL> + <suite de CL> "]"
<PD>             ::= <suite de CL>
<suite de CL>    ::= <CL> + <suite de CL> ! <CL>
<CL>             ::= Nom d'une CL utilisée en présyntaxe.
```

Dans la partie gauche les homographies sont représentées par l'ensemble des CL qui les composent. Cet ensemble est noté entre crochets. La partie droite indique les CL à choisir pour toutes les homographies présentes dans la partie gauche, dans l'ordre où elles apparaisent.

BIBLIOGRAPHIE

(AHO) : Aho A.V. , Ullman J.D.
The theory of parsing, translation and compiling.
Vol I : parsing.
1972, Prentice-hall, inc.

(AN) : Andreewsky A and Fluhr C
Computational learning of semantic lexical relation for the generation and automatical analysis of content.
1977

(CHO) : Chomsky N.
Aspects of theory of syntax.
The MIT Press 1965.

(COLB) : Colby K.M
Conversational langage comprehension using integrated pattern matching and parsing
AI Vol 9, 1977

(COU1) : Courtin J
Un analyseur syntaxique interactif pour la communication homme-machine
Proc JCCL Pise 1973

(COU2) : Courtin J
Utilisation des redondances pour l'analyse et le contrôle automatique des langues.
Proc JCCL Ottawa 1976

(COU3) : Courtin J
Algorithmes pour le traitement interactif des langues naturelles.
Thèse d'état Université de Grenoble 1977

(COUD) : Courtin J. Dujardin D.
Paramètres linguistiques de la morphplogie française dans le système P.I.A.F.
Document interne. Grenoble 1976.

(CUL1) : Culet A.
Analyse de quelques systèmes de "compréhension" d'énoncés en langue naturelle. Analyseur syntaxique du système PIAF.
DEA. Grenoble 1978.

(CUL2) : Culet A.
Application de méthodes d'analyse linguistique à la gestion d'une base de nomenclatures : le logiciel PIAFBCN.
Thèse de Docteur-ingénieur. Grenoble 1981.

(DES) : Descotte Y.
Représentation et exploitation de connaissances "expertes" en génération de plans d'action.
Thèse de 3eme cycle. Grenoble 1981.

(FLOYD) : Floyd R.W.
Bounded context syntactic analysis.
Comm. ACM 7, 1964.

Syntactic analysis and operator precedence. JACM 10, 1963.

(FLU) : Fluhr C
Algorithmes à apprentissage et traitement automatique des langues.
Thèse 1977.

(GRA1) : Grandjean E.
Algorithmes de construction et de mise à jour d'un dictionnaire.
Rapport de recherche No 28.
Université de Grenoble. 1976.

(GRA2) : Grandjean E.
Définitions et utilisation du produit prototype PIAFDOC.
document général à l'usage des documentalistes. 1979.

(GRA3) : Grandjean E.
Traitement des informations textuelles. Projet de recherche PIAF.
Document interne, rapport général PIAF. 1980.

(GRA4) : Grandjean E.
Conception et réalisation d'un dictionnaire pour un analyseur interactif de langues naturelles.
Mémoire CNAM. 1979.

(GRAV) : Grandjean E. et Veillon G.
Utilisation d'une composante linguistique dans les logiciels de recherche d'information. Logiciel prototype PIAFDOC.
Journée AFCET. Avril 1980.

(GRO) : Gross M
Méthodes en syntaxes.
Ed Hermann 1977

(JOL) : Joloboff V
Unification d'arborescences. Evaluation sémantique d'énoncés en langue naturelle.
Thèse de docteur Ingénieur. Université de Grenoble 1978

(KNU1) : Knuth D.E
The art of computer programming.
Volume 1, 3. Addison Wesley 1973.

(KNU2) : Knuth D.E.
Optimum binary search trees.
Acta Informatica - 1971.

(KNU3) : Knuth D.E.
On the translation of languages from left to right. Information and Control.
1965.

(KNU4) : Knuth D.E.
Top down analysis.
International summer school on computer programming.
Copenhague - Aout 1967.

(LOP) : Lopez Medina J.E.
Systèmes de transformation de ramifications paramétrées. Définitions et applications.
Thèse de docteur-ingénieur. Grenoble 1979.

(MAT) : Mathieu B.
Etude et réalisation d'un système permettant la construction de réseaux d'automates d'états finis - application à la production de documents en braille abrégé.
Thèse de docteur-ingénieur. Grenoble 1980.

(MIST) : MISTRAL III / IV, Série IRIS80
Guide d'exploitation.
Description fonctionnelle.
Guide de l'utilisateur.
Guide de mise en oeuvre.
Compagnie CII-Honeywell-Bull. 1978.

(NILS) : Nilson J.N.
Principle of artificial intelligence
Tioga publishing company. PALO ALTO. CALIFORNIA. 1978.

(NILS1) : Nilson J.N.
Problem-solving methods in artificial intelligence.
McGRAW-HILL computer science series. 1971.

(POG) : Pognan P.
Analyse automatique du tchèque. Vers un algorithme de "compréhension" implicite des textes scientifiques. Définition d'un module prédicatif général.
Thèse d'Etat. Université de Paris3- Sorbonne Nouvelle.

(SALT) : Salton G.
The SMART System. Experience in Automatic Document Processing.
Prentice Hall.
Inc. 1971

(SHI) : Shank RC
Identification of conceptualization underlying natural language.
Computer Models of thought and language.
Freeman san Francisco 1973.

(SH2) : Shank RC and RP Abelson
Scripts, plans and knowledge.
IJCAI 1975

(SH3) : Shank R.C. et Colby K.M.
Computer models of thought and langage.
Freeman 1973.

(TES) : Tesnière
Elément de syntaxe structurale.
Klinchsieck 1959.

(VAUQ) : Vauquois B.
La traduction automatique à Grenoble.
Document de linguistique quantitative No 24. DUNOD 1975.

(VEILL1): Veillon G.
Modèles et algorithmes pour la traduction automatique.
Thèse d'état - Université de Grenoble.

(VEILL2): Veillon G.
Algorithmique.
Cours Université de grenoble, 1974.

(WI1) : Wilks Y
Knowledge structures and language boundaries.
IJCAI MIT 1977

(WI2) : Wilks Y
An intelligent analyser and understander of English.
ACM Vol 18 may 1975

(WIN) : Winograd T
Procedures as a representation for data in a computer program for understanding natural language.
Ph.D.Thesis MIT Massachussets 1971.

(WO1) : Woods W.
Transition network grammar for natural language analysis.
ACM vol. 13, No 10, Octobre 1970.

(WO2) : Woods et Al
Speech understanding system.
Final technical progress report B B N
report No 3438, Cambridge 1976.

AUTORISATION DE SOUTENANCE

VU les dispositions de l'article 3 de l'arrêté du 16 avril 1974,

VU les rapports de présentation de Messieurs

. G. VEILLON, Professeur à l'I.N.P.-G

. H. ALARDO, Chef du département logiciel d'application à Télésystèmes-Questel

Monsieur MERLE Alain

est autorisé à présenter une thèse en soutenance pour l'obtention du diplôme de DOCTEUR-INGENIEUR, spécialité "Génie informatique".

Grenoble, le 9 septembre 1982

Le Président de l'I.N.P.-G.

D. BLOCH

Président

de l'Institut National Polytechnique

de Grenoble

P.O. le Vice-Président,