CELINE, vers un correcteur lexico-syntaxique adaptatif et semi-automatique

Jacques Menezo

THÈSE

présentée par

JACQUES MENÉZO

pour obtenir le grade de

DOCTEUR DE L'INSTITUT POLYTECHNIQUE DE GRENOBLE

Spécialité
INFORMATIQUE

Option
SYSTÈMES ET COMMUNICATIONS

N° attribué par la bibliothèque :

CELINE,
VERS UN CORRECTEUR LEXICO-SYNTAXIQUE, ADAPTATIF ET SEMI-AUTOMATIQUE.

---

5 juillet 1999

Directeur de thèse :
M. Jacques COURTIN

Jury :

M.	Augustin LUX	Président
M.	Jacques-Henri JAYEZ	Rapporteur
M.	Guy PÉRENNOU	Rapporteur
Me.	Catherine GARBAY	Examinatrice
M.	Jacques COURTIN	Directeur
M.	Damien GENTHIAL	Co-directeur

THÉSE PRÉPARÉE AU SEIN DU LABORATOIRE CLIPS
(IMAG, INPG, UJF & CNRS)

Remerciements

Je tiens à remercier tout d'abord Monsieur Augustin Lux, Professeur à l'Institut National Polytechnique de Grenoble, qui a spontanément accepté de présider ce jury.

Je remercie très sincèrement Monsieur Jacques-Henri Jayez, Professeur à l'Université de Nantes, et Monsieur Guy Pérennou, Professeur à l'université Paul Sabatier de Toulouse qui ont bien voulu accepter d'être rapporteurs de cette thèse.

Je remercie Madame Catherine Garbay, Chargée de recherches au Centre National de la Recherche Scientifique (Laboratoire TIMC - Techniques de l'Imagerie, de la Modélisation et de la Cognition) qui a bien voulu être examinatrice de cette thèse.

J'exprime toute ma reconnaissance et mon amitié à Monsieur Jacques Courtin, professeur à l'Université Pierre Mendès France, qui a dirigé cette thèse. Ses enseignements, il y a quelques années, m'ont fait me passionner pour l'informatique. Son rayonnement et sa présence chaleureuse de la première année du Conservatoire National des Arts et Métiers à la dernière année de thèse, ont largement contribué à mon désir d'aller jusqu'au bout de cette formation informatique, somme toute un peu tardive !

Je remercie Monsieur Damien Genthial, Maître de conférences à l'université Pierre Mendés France, co-directeur de cette thèse, pour la justesse de ses remarques et dont les conseils m'ont été particulièrement précieux. Je tiens à lui exprimer toute ma reconnaissance pour l'aide qu'il m'a apportée.

Je remercie tout spécialement Danièle Dujardin, ingénieur CNRS, pour sa présence, sa disponibilité, son écoute, son humour et son soutien appliqués aussi bien à des conseils techniques sur le plan de la linguistique qu'à la relecture de cette thèse. Je me permettrais même de la citer : « Un rêve de l'agent DD : voir avant un proche départ à la retraite, une implantation de CELINE gérant une multitude d'agents spécialistes de la correction de thèses écrites en français pouvant le remplacer …».

Je tiens aussi à exprimer ma gratitude à mon chef d'établissement M. Bernard Bayet, qui a accepté avec patience d'optimiser mes emplois du temps, qui n'a jamais eu la moindre remarque pour des absences dues à quelques colloques ou réunions indispensables …

Je ne pourrai pas oublier d'adresser des pensées affectueuses à Bruno, Céline et Delphine qui ont accepté au quotidien et pendant si longtemps un père et un mari souvent absent ou indisponible et perdu dans ses réflexions.

Pour terminer, restant dans la tradition, je remercie pour nos fous-rires, nos bringues et nos conversations passionnées … les fantômes du bâtiment B de l'IMAG, seules entités respectant fidèlement mes horaires de travail des nuits et des week-ends.

TABLE DES MATIERES

1. INTRODUCTION

Ce chapitre d'introduction est divisé en deux grandes parties :
Une première partie d'approche de la correction d'erreur dans laquelle, à travers quelques faits de base concernant le **T**raitement **I**nformatique des **L**angues **N**aturelles (TALN) et la correction des erreurs, nous allons progressivement voir se préciser des points clés comme

- le besoin de spécifier le rédacteur en vue d'une optimisation de l'heuristique mise en jeu,
- ou sur le plan logiciel, l'intérêt d'une architecture d'abord multi-modules puis multi-agents

Une deuxième partie nous présentera le plan de cette thèse, plan induit par une progressivité dans la satisfaction des exigences.

1.1 UNE APPROCHE DE LA CORRECTION D'ERREURS

1.1.1 PREMIÈRE ÉTAPE : LE MULTI-MODULES

L'équipe TRILAN m'avait conquis à l'occasion de la préparation du mémoire d'ingénieur en informatique, tout d'abord par sa composition (Jacques COURTIN, Damien GENTHIAL, Danièle DUJARDIN, ...) mais aussi par le sujet de mémoire proposé : un développement logiciel s'appuyant sur un travail statistique dans le domaine du dialogue oral. La partie linguistique étant confiée à une linguiste Anne-Lyse FRECHET (Frechet 93), l'informaticien mathématicien s'occupait des statistiques et des développements logiciels (Menézo 92), chacun était à sa place et tout était pour le mieux dans le meilleur des mondes...

C'est donc sans méfiance particulière, que j'abordais cette thèse sur le thème de la correction des erreurs. Les termes informatiques m'appâtèrent : explorer les systèmes répartis, les systèmes multi-agents, introduire de l'intelligence artificielle ! La linguiste avait disparu mais naïvement je me disais qu'avec un bon lexique, quelques braves règles grammaticales bien écrites et l'analyseur syntaxique adéquat, on devait pouvoir s'en sortir, et que là n'était pas le problème. Le désenchantement, le désespoir le plus noir vinrent vite : le bon lexique n'existait pas ! Les braves règles grammaticales et l'analyseur syntaxique non plus !

Pire ! Je m'aperçus que la simple analyse en langue naturelle d'un texte écrit non entaché d'erreurs n'était pas près d'être résolue. La compréhension du texte, indispensable pour diminuer le degré d'ambiguïté, demandait l'intervention de traits sémantiques, traits absents des grands lexiques. Il n'y avait pas de lexique unique mais des lexiques par domaine et une bonne couverture lexicale demandait d'appliquer des règles grammaticales ou sémantiques spécifiques au domaine traité. Les connaissances étaient fragmentées, éparpillées dans différentes équipes, il n'y avait pas de formalisme commun etc. Je compris alors ce qu'était un système réparti ! Le système réparti arrivait tout doucement sous une facette multi-lexiques, multi-grammaires, multi-sémantiques, bref multi-modules. Ces modules devraient fournir leurs données, travailler ensemble, collaborer à un but commun et leurs modes d'interactions leur permettraient peut-être de mériter le qualificatif distingué et recherché d'agents.

De plus, il ne s'agissait pas simplement d'analyser un texte bien écrit ! « Mes textes » contenaient potentiellement des erreurs ! La reconnaissance du sans faute étant déjà difficile pour ne pas dire impossible, comment pouvait-on reconnaître un mot faux et en outre le corriger ! La non-reconnaissance lexicale ne semblait pas une notion suffisante compte tenu de la fragmentation de la langue en domaines.

En tout cas, pour faire face à cet état peu avancé de l'art, je posais en postulat une hypothèse forte : ces lexiques existaient, incomplets ou pas, inexacts ou pas. Je devais les utiliser en les prenant pour ce qu'ils étaient ! L'aspect pseudo-intelligent pourrait peut-être provenir de la façon de gérer le travail sur un ensemble de lexiques placés en situation de concurrence.

La question « lexiques et ensembles de règles grammaticales ou sémantiques s'y rattachant » étant supposée réglée, il y avait lieu pour déblayer le sujet, de regarder d'un peu plus près la correction des erreurs. Nous en profiterons pour revenir au présent. Dans les deux étapes suivantes, nous allons « redécouvrir » que la correction d'erreurs demande non pas de travailler sur la graphie d'un mot isolé mais de

- prendre en compte le contexte morphologique local,
- mener en parallèle analyse lexicale, analyse syntaxique et analyse sémantique,
- diminuer le caractère ambigu des analyses en travaillant dans des domaines particuliers (lexiques et grammaires locales),
- tenir compte des habitudes du rédacteur par une sélectivité de l'ensemble des règles heuristiques applicables en ciblant l'efficacité d'une correction automatique,
- introduire une composante statistique permettant de suivre une évolution de la langue utilisée ou des compétences du rédacteur en ciblant l'optimisation d'une correction interactive.

En conséquence nous pourrons mieux entrevoir les exigences en termes d'architecture du système de correction des erreurs.

1.1.2 DEUXIÈME ÉTAPE : LE BESOIN DE PARALLÉLISME

1.1.2.1 Prise en compte du contexte

Faisons un retour en arrière de cinq ans correspondant à la durée de cette thèse et « cliquons » sur la rubrique *outils* du traitement de texte mondialement le plus utilisé WORD (version française 96 PC 6.0x). Le menu déroulant nous propose alors deux rubriques : orthographe suivie de grammaire c'est à dire vérification lexicale avec éventuellement proposition(s) de correction et analyse « syntaxique » avec vérification des accords en nombre en genre et en temps. Comme la majorité des utilisateurs de l'époque, nous apprécierons la vérification lexicale mais nous n'utiliserons pas la vérification grammaticale, trop lente, crispante et, de plus, souvent douteuse ou complètement farfelue.

Cinq ans après, le correcteur orthographique de WORD est devenu plus performant et agréable à utiliser, sans doute globalement plus efficace mais on trouve très vite certaines limites. Considérons un exemple précis en comparant les propositions de

correction par WORD 6 et WORD 97 (version de janvier 98) pour les trois groupes nominaux :

1. *les jolis chevas,*
2. *le joli chevax*
3. *les jolis chevax.*

Groupes nominaux	WORD 6		WORD 97	
	Solution(s) proposée(s)	Commentaire	Solution(s) proposée(s)	Commentaire
Les jolis chevas	*cheval*	Proposition au singulier Pas de prise en compte ni du *s* ni du contexte *les jolis*	Pas de proposition !	?
Le joli chevax	*cheval* *ou* *chevaux*	Proposition du singulier et du pluriel Pas de prise en compte ni du *x* ni du contexte *le joli*	*chevaux*	Cette fois le x a été pris en compte comme marque du pluriel puisque WORD ne propose plus cheval Pas de prise en compte du contexte *le joli*
Les jolis chevax	*cheval* *ou* *chevaux*	Proposition du singulier et du pluriel Pas de prise en compte ni du *x* ni du contexte *le joli*	*chevaux*	Comme ci-dessus

Conclusion :

- WORD 6 ne s'autorise la substitution, la suppression ou l'ajout que d'une seule lettre (choix d'une règle heuristique sur la distance de correction). De ce fait *chevas* ne peut donner que *cheval* alors que *chevax* peut donner *cheval* par substitution et *chevaux* par ajout. Il n'y a ni prise en compte du contexte ni règle heuristique considérant le *s* comme marque du pluriel.
- WORD 97 prend en compte totalement la marque du pluriel amené par le *x* final puisqu'il ne propose plus *cheval* ou *chevaux* mais uniquement *chevaux*, par contre le contexte amené par *le joli* ou *les jolis* n'est toujours pas pris en compte. Nous pouvons en déduire que l'heuristique sous-jacente a changé mais nous restons toujours insatisfaits.

1.1.2.2 Insuffisance du traitement séquentiel

La quasi-totalité des correcteurs utilisait et utilise toujours d'ailleurs en séquence : un traitement lexical et un traitement syntaxique (figure 1.1). Pour reprendre le cas de WORD 97, nous pourrions avoir un espoir puisque, à la place des deux rubriques *orthographe* et *grammaire*, nous n'en trouvons qu'une seule : *grammaire <u>et</u> orthographe*. Malheureusement, l'examen du fonctionnement nous a montré que le traitement séquentiel reste toujours en vigueur.

1.1.2.2.1 Au niveau lexical

Chaque mot est analysé. En cas de mots inconnus (ce qui ne veut pas dire erreurs), des propositions de mot remplaçant sont éventuellement faites. Selon les logiciels, ces propositions sont élaborées en utilisant diverses méthodes (clé squelette, clé phonétique, traitement alphabétique, génération morphologique) mais en partant presque exclusivement de la graphie du mot inconnu. L'application de règles heuristiques permet de limiter le nombre de solutions concurrentes.

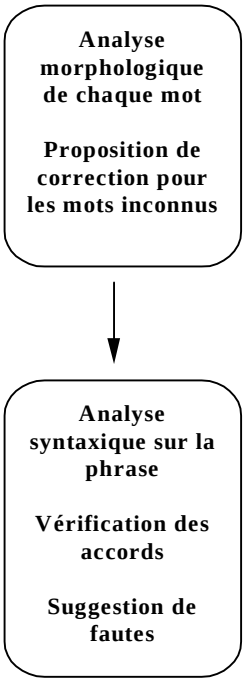


Figure 1.1 : Traitement séquentiel

Pour illustrer l'insuffisance de cette source de connaissance, prenons le cas de la génération morphologique. La génération morphologique (Courtin 91) permet de constituer le mot correct à partir

- de la ou des racine(s) supposée(s) du mot,
- des variables morphologiques associées déduites des désinences telles que nombre et genre pour un substantif, groupe, personne, nombre, mode et temps pour un verbe.

Il est bien évident que *la levée d'ambiguïté d'un mot inconnu demande le maximum d'informations* et que la simple observation de la graphie inconnue n'est pas suffisante. Les informations manquantes (ou incertaines) sur la ou les racines possibles ainsi que sur les variables morphologiques peuvent être obtenues par d'autres étapes des analyses.

Par exemple :

1. Une analyse syntaxique attribuant au groupe nominal *les jolis chevax* le nombre 'pluriel' permet de choisir plus sereinement *chevaux* et renforce la marque du pluriel amené par le *x*.
2. Certains analyseurs syntaxiques (ex : analyseur syntaxique de Earley, analyseur syntaxique par la méthode des CHARTS) ou

une modélisation multi-niveaux par des HMM (modèles cachés de MARKOV) peuvent proposer une catégorie morphologique pour un mot inconnu.

3. Un module de correction des erreurs peut proposer les variables *nombre, genre, temps* pour un verbe

1.1.2.2.2 Au niveau syntaxique

La simple correction d'un mot va peut-être changer l'analyse syntaxique (par exemple : si cette correction change la catégorie morphologique), un retour arrière s'impose alors pour tout reprendre. Compte tenu du haut degré d'ambiguïté, on obtient alors une explosion combinatoire à laquelle on échappe par des règles heuristiques très restrictives de limitation des solutions. On réalise alors qu'il faudrait un traitement simultané des différentes phases de l'analyse avec des interactions des différents mots ou qu'à défaut il faudrait mener le traitement non pas phrase par phrase mais groupe de mot par groupe de mot.

1.1.2.2.3 Conclusion : système dynamique de correction des erreurs

L'analyse lexicale seule ne permet pas de prendre des décisions et elle doit s'appuyer sur les résultats de l'analyse syntaxique qui a besoin de l'analyse lexicale ! Nous pouvons en faire une interprétation moderne : il s'agit d'un système dynamique par ses interactions.

Un fait incontournable apparaît : face à une erreur potentielle, il faut retarder la prise de décision et faire intervenir en parallèle des résultats partiels obtenus sur différents niveaux. Pour diminuer le niveau d'ambiguïté il y a lieu d'appliquer si possible des règles heuristiques

1.1.2.3 Contexte et interaction des niveaux de correction

L'analyse de notre insatisfaction a montré qu'elle trouvait son origine dans une absence de prise en compte du contexte. La prise en compte du contexte demande d'analyser ce contexte non seulement d'un point de vue lexical mais aussi d'un point de vue grammatical.

Pour illustrer cette démarche nous examinerons la correction partielle du groupe nominal dans la phrase [1] :

« *le baeu chevax noir racée et les juents courrent furieux et affamée* »

comportant diverses fautes orthographiques ou d'accords. La reconnaissance lexicale va buter sur trois mots inconnus à corriger. Le système de correction va proposer des solutions en fonction des heuristiques choisies.

- Par exemple *baeu* est corrigible en *beau* (*subc* substantif commun ou *adjq* adjectif) ou en *bau* (*subc*) si on suppose une seule faute par mot, mais aussi en *beaux* (*subc* ou *adjq*) ou en *bague* (*subc*) si on accepte deux fautes par mots, etc. Nous observons sur ce cas particulier l'intérêt du système à connaître l'utilisateur pour faire un choix dans les distances de correction. Supposons pour simplifier que la règle heuristique adoptée soit celle d'une seule faute par mot.
- *chevax* est alors corrigible en *cheval* ou en *chevaux.*
- *juents* est alors corrigible en *juments*.
- La prise en compte de la règle heuristique dite de « la flemme du rédacteur », se traduisant dans ce cas par *on ajoute rarement une marque de pluriel mais on les oublie fréquemment*, permet de corriger initialement *chevax* en *chevaux* en considérant le *x* comme une marque du pluriel (détection par traitement des suffixes).

le	*baeu*	*chevax*	*noir*	*racée*
det[2]	subc	subc	subc	adjq
	adjq		adjq	

- Plaçons nous au niveau des catégories morphologiques et considérons les cinq premiers mots. Les modèles cachés de Markov et une matrice de transition (spécifique de l'utilisateur ou par défaut) vont aider au choix entre *bau* et *beau* (utilisation d'un modèle tri-classe avec probabilité élevée des transitions *det adjq subc* au-dessus du seuil de confiance et probabilité quasi nulle ou très faible des transitions *det subc subc* et *det subc subc subc*) (Kallas 87) (Menézo 92). La séquence *le beau chevaux (det adjq subc)* est retenue et la séquence *le bau chevaux (det subc subc)* est rejetée. Il va en être de même pour le rejet de *subc* pour le mot *noir*. Nous observons donc qu'un traitement markovien peut aider :
  1. la correction lexicale,
  2. la correction syntaxique en permettant une levée d'ambiguïté, avec pour solution finale retenue, l'enchaînement *det adjq subc adjq adjq*.

[1] Phrase tirée d'une rédaction avec traitement de texte d'un élève de quatrième. Cette phrase a été reprise en changeant les adjectifs (par exemple *noir* masculin/singulier remplacé par *marron* invariant) de manière à tester les conséquences des substitutions.
[2] det : déterminant (un traitement avait exclu la catégorie *pronom*).

- L'analyse syntaxique après la désambiguïsation syntaxique précédente va permettre de reconnaître un groupement nominal pour le groupe des cinq premiers mots.
- Une vérification des accords en nombre sur ce groupe nominal va permettre de ne retenir que *cheval* comme correction de *chevax*.
- En parallèle, la correction de *chevax* en *cheval* avec le rejet de *chevaux* a pour conséquence de rendre douteuse, pour cet utilisateur, l'application de la règle de la flemme du rédacteur.

Récapitulons : la correction de *chevax* va demander ou entraîner au moins neuf interventions :

1. une analyse morphologique (détermination de la racine *chev*),
2. une génération morphologique (production à partir de la racine de *chevaux* en prenant en compte le signe du pluriel amené par le *x*),
3. une désambiguïsation grammaticale (sélection de l'enchaînement *det adjq subc adjq adjq*),
4. une analyse syntaxique (reconnaissance du groupe nominal),
5. une vérification des accords en nombre (unification au singulier),
6. une génération morphologique (*chevaux* en *cheval*).

avec simultanément un déclenchement de correction pour un autre mot :

7. une vérification des accords en genre (unification au masculin),
8. une génération morphologique (*racée* en *racé*),

et sur un autre plan

9. une prise en compte d'un exemple prenant en défaut la règle de la flemme du rédacteur, prise en compte pouvant changer l'heuristique et correspondant à une acquisition d'expérience par le système.

Concluons et généralisons :

- Nous redécouvrons donc (Courtin 89) (Letellier, 93) (Stéfanini 93) (Genthial, 94) que la correction d'erreurs demande de prendre en compte simultanément un ensemble d'indices correspondant à des traits lexicaux, syntaxiques, sémantiques, pragmatiques, phonétiques... De ce fait, le traitement présente plusieurs phases mais demande, pour être optimisé ou même simplement efficace, non pas un ordonnancement séquentiel de ces phases mais un traitement en parallèle.

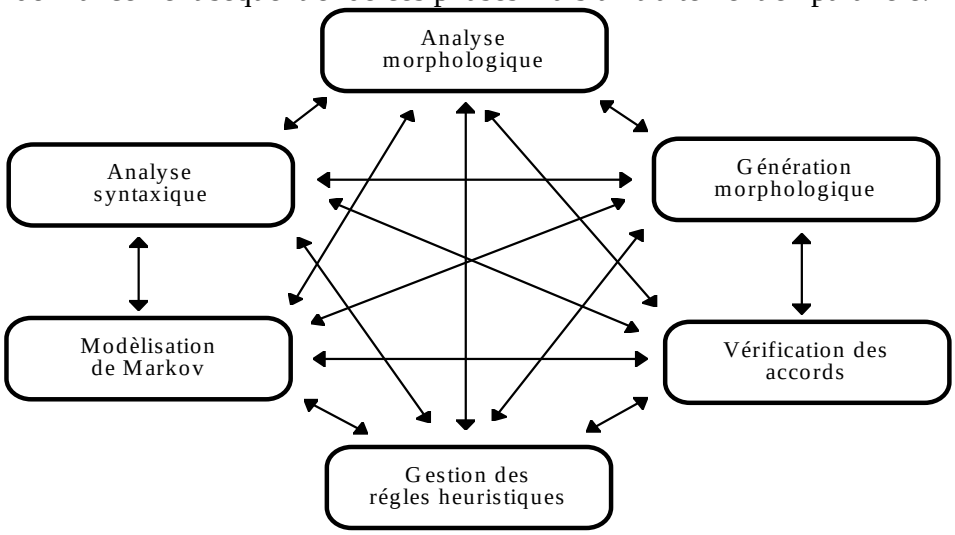


Figure 1.2 : Besoins d'échanges d'informations

- Par ailleurs, il est fondamental de pouvoir travailler sur des ensembles à faible cardinalité de mots, chaque groupe constituant un contexte pour les groupes voisins (adjacents ou non).
- Il est tout aussi fondamental d'avoir une heuristique adaptée à l'utilisateur. Cette adaptation peut être acquise au fil des corrections.
- Nous sentons le besoin de communication inter-traitements (inter-modules) ce qui correspond à un schéma dit "hétérarchique" des communications (figure 1.2).

1.1.2.4 Ambiguïté des langues naturelles et conclusion de l'étape 2

Les paragraphes précédents ont introduit la notion d'ambiguïté emmenant de ce fait à un ensemble de solutions multiples dont le cardinal pouvait être réduit par application de règles heuristiques.

1.1.2.4.1 La nature des connaissances.

Les langues naturelles présentent un haut degré d'ambiguïté. Ce degré est largement fonction du caractère général ou au contraire spécifique du mode de représentation des connaissances que ce soit au niveau lexical ou au niveau syntaxique. Plus le mode de représentation est général et plus le nombre de solutions parasites ou d'ambiguïtés engendrées par le système est grand.

- Prenons un exemple au niveau lexical : le mot *actions* peut être soit le substantif féminin pluriel bien connu soit une forme conjuguée du verbe *acter* qui n'est utilisé que dans le domaine juridique et donc fort rarement par la majorité des textes. Il semble donc inutile à chaque fois qu'un texte contient la graphie *actions,* d'avoir à traiter l'ambiguïté de catégorie morphologique substantif ou verbe.
- Au niveau syntaxique là encore, l'écriture d'une grammaire la plus générale possible va induire, par interférence entre règles, un maximum de solutions parasites, solutions du système de règles mais ne tenant pas compte des habitudes réelles de parole.

Il semble donc intéressant de distinguer sur le plan lexical un dictionnaire de base et des dictionnaires de spécialité et sur le plan syntaxique une grammaire générale et des grammaires locales. De ce fait, on aboutit tout naturellement à des modules (... des agents) spécialisés. (Stéphanini 93) consacre une soixantaine de pages, à cette problématique linguistique de l'analyse du français écrit avec de nombreux exemples.

1.1.2.4.2 La nécessité d'une collaboration

Comme nous l'avons déjà rappelé, la correction d'erreurs présente plusieurs phases (lexicale, syntaxique, sémantique, pragmatique, phonétique...) et demande pour être optimisée ou même simplement efficace, non pas un ordonnancement séquentiel de ces phases mais un traitement en parallèle :

- Les ambiguïtés des langues naturelles, encore augmentées en présence d'erreurs, interdisent pratiquement qu'une seule des phases de l'analyse aboutisse, en ce qui concerne les traits manipulés par cette phase, à une solution unique désambiguïsée. Une vérification correction complète va demander une collaboration de l'ensemble des traits pertinents disponibles, à travers les différentes phases, pour éliminer le maximum de solutions concurrentes.
- Chaque étape (échec ou réussite) d'une des phases peut être riche de conséquences pour une autre. On éliminera peut-être des solutions

concurrentes ou au contraire on pourra proposer des solutions non encore perçues.

- La recherche de toutes les solutions ou la rencontre d'impasses lors d'une des phases entraînent des retours en arrière sur les hypothèses émises dans d'autres phases, hypothèses encore inexploitées. Le coût global du traitement diminue lors d'un traitement en parallèle.

1.1.3 TROISIÈME ÉTAPE : LES INTERACTIONS ET LE MULTI-AGENTS

1.1.3.1 Architecture d'un système de correction des errreurs

La première étape de cette approche a introduit la notion de modules, la deuxième étape a amené la nécessité d'utiliser les connaissances de ces modules par des traitements interactifs, en parallèle. Subséquemment, concernant le choix d'une architecture d'un système de correction d'erreurs lexico-syntaxique deux solutions extrêmes sont possibles :

1. On peut concevoir un logiciel monolithique dans lequel les différentes phases seraient confondues, le logiciel disposant de l'ensemble des informations utiles à partir d'une base de données unique regroupant, sous une forme ou une autre, tous les traits nécessaires (lexicaux, grammaticaux, sémantiques phonétiques, styles etc.). Cette solution qui permet une prise en compte simultanée de toutes les phases de l'analyse semble très prometteuse et à terme peut-être même indispensable. Pourtant nous l'écarterons a priori pour une raison contextuelle majeure : aucune modélisation suffisamment avancée de ce type n'a vu le jour et il n'existe pas de bases de données de ce type. Nous verrons au chapitre 8 sur les systèmes multi-agents que l'architecture dite *tableau noir* permet une simulation de ce fonctionnement.

2. A l'opposé de cette conception centralisée, on peut aussi envisager un système réparti d'agents spécialisés possédant chacun ses connaissances et collaborant à cette tâche. La problématique de l'analyse des langues naturelles et encore plus la problématique de la correction d'erreurs vont nous amener, compte tenu de l'état de l'art contextuel, à choisir ce type d'architecture répartie. Sur le plan du génie logiciel, un avantage important qui en découle, réside dans la souplesse de développement comparable à celle de la classique décomposition modulaire. Chaque équipe peut continuer à faire évoluer ses bases de connaissances et ses agents de manière quasi-indépendante, la seule contrainte obligatoire étant le respect du mode de communication choisi et des protocoles s'y rattachant. Si un des inconvénients de ce choix est de maintenir partiellement le traitement séquentiel par phase, par contre, et cela représente quand même un pas important, la granularité va être bien plus fine puisqu'on va passer, au niveau syntaxique, d'un traitement au niveau de l'ensemble de la phrase à un traitement au niveau de groupes de mots que nous appellerons *structures*.

Remarque : conceptuellement l'architecture du premier type contient en soi la deuxième et vice versa à condition d'envisager chaque mot comme un agent particulier d'une société d'agents qui interagit avec ses voisins plus ou moins proches. Une démarche équivalente a été proposée, par exemple, pour un système multi-agents de traitement des cartes de géographie, chaque point de la carte étant un agent (SMA et IAD Chambéry 95). Ce qui est possible pour les points d'une carte de géographie dont les attributs sont

clairement définis, ne l'est pas pour les mots (absence de modélisation complète précédemment évoquée).

1.1.3.2 Démarche de correction et architecture multi-agents

Comme l'avons vu, le choix d'une architecture multi-agents peut être justifié par les structures de données disponibles, mais sa véritable justification réside dans la démarche de recherche d'informations en vue de déterminer la correction la plus pertinente possible.

Citons Jacques Ferber (Ferber 1989) : "*Les systèmes multi-agents apportent bien plus que des découpages de connaissances : ils renouvellent notre façon d'aborder un problème et de concevoir ce qu'est le raisonnement et l'intelligence. Le problème ne se résout plus en partant d'un état initial pour arriver à un état final, mais en construisant au fur et à mesure des solutions partielles qui se trouvent sur le chemin, chaque agent cherchant à apporter des éléments de solution au fur et à mesure de l'exploration"*. Outre le découpage des connaissances, cette vision correspond exactement à la correction des erreurs telle qu'elle est appliquée dans le système proposé CELINE (Menézo 96 a) (Menézo 96 b) (Menézo 96 c). Cette architecture permet de mettre un certain nombre d'agents en concurrence pour la majorité des prises de décision. Le contexte de décision d'abord local va ensuite être étendu à l'ensemble de la phrase et va souvent être multi-niveaux, c'est-à-dire faire intervenir aussi bien la graphie de la forme de surface que les formes cachées telles que catégories morphologiques, variables morphologiques, structures syntaxiques.

Une difficulté importante réside dans l'exigence de pouvoir construire des solutions partielles en amenant des renseignements multi-niveaux sur un mot ou des groupes de mots sans traiter la totalité de la phrase. Comme nous le verrons, le Modèle Linguistique Partiel Suffisant (MLPS) va nous permettre d'atteindre cette finesse de granularité dans la manipulation des informations.

1.1.3.3 Exemple de coopération entre agents

Il est donc temps de formaliser, au moins sur un exemple particulier, quelles peuvent être ces interactions. Nous souhaiterons alors un travail intelligent des modules ce qui permettra de leur donner le terme d'agents.
Dans la terminologie des architectures multi-agents (S.M.A.) et de l'intelligence artificielle distribuée (I.A.D.), les agents sont organisés en société, et de ce fait, par ses connaissances et ses relations, chaque agent possède un comportement social, comportement qui tend dans certaines réalisations de l'I.A.D. à reproduire certains aspects du comportement humain. Les connaissances et le comportement de l'agent peuvent être évolutifs. Chaque agent peut posséder un aspect cognitif plus ou moins marqué ou au contraire être purement réactif. L'intelligence du système émerge de l'activité globale résultant de la collaboration entre agents et provient aussi de l'évolution des connaissances et de l'expérience de chaque agent.

Pour illustrer cette activité sociale, le scénario de correction envisagé pour la phrase : « *le baeu chevax noir racée et les juents courrent furieux et affamée* » permet de suivre ce besoin de collaboration et présente quelques négociations entre agents :

1. Pour la correction de *baeu* un pilote des activités lexicales (PAL) interroge un agent d'analyse morphologique (PILAF) qui fournit deux solutions *beau* (adjectif ou substantif) et *bau* (substantif, largeur d'un bateau).

2. Pour la correction de *chevax*, le pilote PAL et l'agent PILAF trouvent une solution *chevaux* (nous avons déjà explicité au §1.1 la collaboration syntaxique sous-jacente).
3. De la même façon, pour la correction de *juents*, le pilote PAL et l'agent PILAF trouvent une solution *juments.*
4. Un agent (PSCM, proposition statistique de catégories morphologiques) utilisant les modèles de Markov avec une matrice de transition par défaut, peut aider au choix entre *bau* et *beau* (utilisation d'un modèle triclasse avec probabilité élevée des transitions *det adjq subc* au-dessus du seuil de confiance et probabilité plus faible de la transition det subc subc).
5. Un pilote des activités syntaxiques (PAAS) et des agents d'analyse syntaxique retiennent comme choix principal la séquence *le beau chevaux (déterminant adjectif substantif)* et garde en réserve *le bau chevaux (déterminant substantif substantif).*
6. Un pilote (PAVA) de l'activité de vérification des accords et un agent (DCFA) de détection et correction des fautes d'accord corrigent les accords en utilisant une méthode particulière dite *méthode des structures*.
7. Un générateur morphologique PILAF fournit les graphies correctes des mots corrigés.
8. Optionnel : un agent FORME formate le texte avec un seul blanc entre deux mots consécutifs et ajoute une majuscule en début de phrase.

Au total la phrase est correctement corrigée en

« *Le beau cheval noir racé et les juments courent furieux et affamés* ».

1.1.3.4 Évolution du système et ajouts de nouveaux agents

On peut ajouter à volonté de nouveaux agents ce qui est particulièrement intéressant dans les domaines lexical et syntaxique pour lesquels de nombreuses équipes de recherche travaillent sur un point particulier. La prise en charge d'un nouveau agent ne demande pas une modification du système déjà en place :

- On ajoute le nom de l'agent et ses coordonnées à la liste des agents éligibles.
- Au niveau heuristique, la mise en œuvre est immédiate par initialisation à des valeurs par défaut des divers coefficients.
- Par contre l'écriture de l'interface est un peu plus exigeante. Il faut déterminer des tables de correspondance entre les traits manipulés par le nouvel agent et les traits en vigueur dans CÉLINE. Dans l'hypothèse de nombreux agents, il n'est pas indispensable que la correspondance des traits soit totale ou très rigoureuse, la complémentarité entre agents fait qu'un renseignement même fragmentaire amené par un agent peut être utile pour la levée des ambiguïtés.

1.1.4 ASPECTS QUANTITATIFS DU T.A.L.N.

Les trois étapes de l'introduction nous ont montré le besoin de règles heuristiques comme aide de la désambiguïsation. Les règles heuristiques peuvent avoir une inspiration intuitive mais leur validité reposent sur une enquête statistique.

L'introduction du double volume *« Traitements Probabilistes et Corpus »* de l'Association pour le Traitement Automatique des langues (ATALA) publié au premier trimestre 1995 reprend un historique des changements de cap de la linguistique informatique *qui connaît un récent engouement pour les approches statistiques et*

probabilistes ainsi que pour celles qui marient symbolique et quantitatif. Citons M. Liberman rappelant le courant anti-empirique, anti-numérique et pro-symbolique des vingt dernières années : « Compter était précisément considéré comme n'étant pas une tâche appropriée pour une personne de qualité » Mais le vent souffle désormais au quantitatif. L'examen des récentes livraisons de Computational Linguistics, comme celui de congrès comme ACL (Association for Computational Linguistics), Coling etc. est éloquent. Une partie des causes est claire : les données disponibles ont changé et d'échelle et de nature (Échelle : on est passé de corpus de quelques centaines de mots à des corpus de plusieurs dizaines de millions de mots. Nature : aux corpus étiquetés des années 80 succèdent des corpus arborés). Les augmentations en capacité de stockage et en puissance de calcul rendent en outre possibles aujourd'hui des traitements inenvisageables hier. Cette tendance au quantitatif est aussi soutenu dans le cadre dit des Industries de la langue par le souci d'obtenir une évaluation des performances des systèmes industriels. (ATALA 95)

Les règles heuristiques mises en jeu dans CELINE reposent généralement sur des connaissances acquises par le biais de statistiques. Leur validité est quantifiée par un ensemble de coefficients. L'aspect quantitatif va intervenir sur huit plans avec des interactions interniveaux :

- Au niveau lexical
  1. Pour une optimisation de l'identification des mots (exemple : choix du « bon lexique »).
  2. Pour le choix d'heuristiques face à un mot inconnu (exemples : connaissance des fautes lexicales de l'utilisateur, modélisation de Markov pour le choix d'une catégorie morphologique).
- Au niveau syntaxique
  3. Pour une optimisation dans le choix du « bon analyseur ».
  4. Pour certains analyseurs dans le choix d'une grammaire stochastique et bien évidemment dans la grammaire utilisée.
  5. Pour le choix d'heuristiques face à un « enchaînement » inconnu (exemples : connaissance des fautes syntaxiques de l'utilisateur, utilisation des modèles de Markov pour le choix d'une catégorie morphologique).
- Au niveau de la correction des erreurs
  6. Pour une optimisation dans le choix du « bon correcteur ».
  7. Pour le choix d'heuristiques face à un « enchaînement » inconnu (exemples : connaissance des fautes courantes de l'utilisateur).
  8. Par des coefficients de pondération pour l'application des règles de corrections des erreurs.

1.1.5 MODÈLE HUMAIN ET MODÈLE LINGUISTIQUE PARTIEL SUFFISANT

La réalisation présentée est sous-tendue par un modèle humain intelligent (*le maître d'école*) car la correction des erreurs est d'autant plus efficace que :

- Le système de correction (*le maître*) possède une expérience personnelle (acquisition par déduction de connaissances nouvelles), les connaissances générales ainsi acquises étant indépendantes du type de texte ou de l'utilisateur (*quel que soit l'élève*).
- Le système de correction (*le maître*) possède une connaissance différenciée par utilisateur (*connaissance de l'élève*). Cette connaissance différenciée est obtenue par des statistiques (*observation des productions de l'élève*).

Si nous nous plaçons au niveau d'un utilisateur particulier, celui-ci

- n'a besoin en général que d'un lexique réduit sous la forme d'un dictionnaire général de faible volume augmenté de quelques domaines particuliers,
- va reproduire sur le plan grammatical toujours les mêmes formes syntaxiques ce qui explique le rendement exceptionnel des modèles de Markov.
- produit annuellement un volume en général négligeable devant les tailles des corpus d'apprentissage mentionnées ci-dessus. Par exemple, pour un chercheur, dix articles d'une dizaine de pages et un mémoire de 100 pages correspondent à un ordre de grandeur d'environ 0,5 Mégaoctets (2500 caractères par page). La production moyenne des utilisateurs de traitement de texte est très inférieure à 0,1 Mégaoctets.

Le créateur d'un système de correction d'erreurs se trouve donc en face d'un double impératif :

- disposer d'un système linguistique virtuel le plus large possible pouvant potentiellement satisfaire n'importe quel utilisateur,
- extraire de ce système un sous-système nécessaire et suffisant pour un utilisateur donné : *le modèle linguistique partiel suffisant* (M.L.P.S.).

La définition du modèle linguistique partiel suffisant comprendra deux étapes :

- Le traitement de l'aspect lexical sous la forme de l'apprentissage des domaines de l'utilisateur permettant la construction d'une table de référence constituant l'aspect lexical du MPLS.
- Le traitement de l'aspect syntaxique sous la forme de la définition d'une grammaire personnalisée ne comprenant qu'un nombre fini de dérivations possibles.

En d'autres termes l'utilisateur doit se retrouver en face *d'une image personnalisée du système de correction* (figure 1.3). Notre proposition va inclure cette création d'image personnalisée à travers un apprentissage automatique et un dialogue entre deux agents : le maître d'école (le système de correction des erreurs) et l'élève (l'utilisateur du système de correction des erreurs).

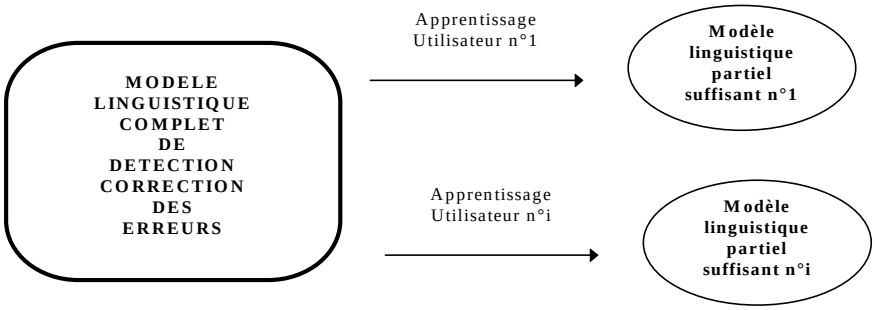


Figure 1.3 : Image virtuelle et image réelle du système

La conception du MLPS va coïncider avec une problématique de l'architecture du système nous conduisant vers une architecture multi-agents.

1.1.6 INTERFACE HOMME-MACHINE ET AUTOMATISME DES CORRECTIONS

A terme, la finalité est de pouvoir faire de la correction d'erreur complètement automatisée comme le ferait un correcteur humain.
De même que dans WORD, on peut choisir les affichages *Normal, Page, Lecture à l'écran, Plan, Document maître,* nous proposons un affichage *Correction globale interactive* dont nous donnons ci-dessous un exemple (figure 1.4) :

Phrase initiale	N°	Propositions de phrases corrigées	Poids
Je manje une lomme	**01**	***Je mange une pomme***	**0.9200**
	02	***Je mange une gomme***	**0.4534**
	03	***Je mange une tomme***	**0.4234**
	04	***Je mande une pomme***	**0.2128**
	05	***Je mande une gomme***	**0.2001**
	06	***Je mande une tomme***	**0.1870**
	07	***Je manie une pomme***	**0.1600**
	08	***Je manie une gomme***	**0.1600**
	09	***Je manie une tomme***	**0.1600**
	…	…	**…**
Corrigez la phrase initiale ou Double cliquez sur la correction sélectionnée			

Le système corrige correctement actuellement 84 phrases sur 100.

: Passage au mode interactif de correction

: **Passage au mode automatique de correction**

: **Traitement de la phrase suivante**

: **Traitement de la phrase précédente**

Un premier objectif intermédiaire déjà ambitieux pourrait être une correction en trois temps :

- Après une première écriture du texte par le rédacteur, on procède à une correction interactive (figure 1.4). On gagne ainsi du temps sur la détermination des domaines, certaines structures, etc.
- Travail de reprise de cette première version du texte : correction automatique.
- Dernier travail sur le texte : validation par l'agent humain de toutes les corrections décidées par le système.

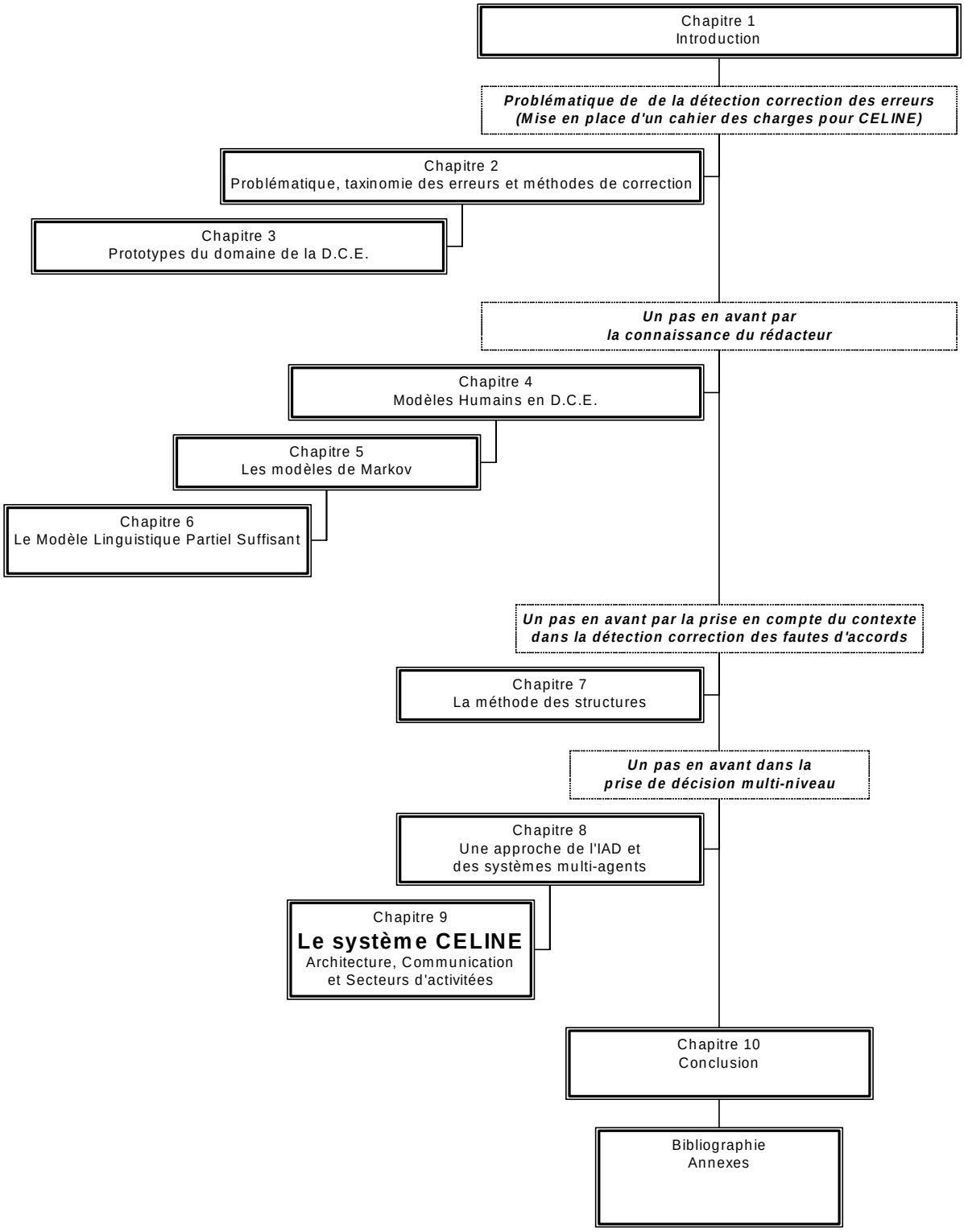
1.2 INDUSTRIE DE LA LANGUE ET NOUVELLES TECHNOLOGIES

L'industrie de la langue commence à envisager les profonds bouleversements des modes de fonctionnement introduit par le réseau INTERNET. De plus en plus, on peut pressentir que les postes de travail futurs seront des terminaux intelligents c'est-à-dire que, bien que dotés d'autonomie, ils resteront liés à un ordinateur central distributeur de ressources (logiciels, banque de données etc).

CELINE s'intègre parfaitement dans cette problématique. Le chapitre de conclusion nous permettra en partant du résumé des choix retenus de situer notre proposition dans un environnement moderne ou futuriste.

1.3 PLAN DE LA THÈSE

1.3.1 PLAN GÉNÉRAL



1.3.2 PLAN DU SPÉCIFIQUE SUR CELINE

Cette thèse est à la croisée de divers domaines :

- l'analyse des langues naturelles,
- la correction des erreurs,
- le parallélisme et l'intelligence artificielle,
- les systèmes multi-agents,

saupoudrés d'une couche de probabilités et de statistiques.

Envisageant un hypothétique lecteur, on peut penser qu'il ne dispose pas forcément du pré-requis spécifique à chacun de ces domaines. Pour chaque domaine, j'ai donc choisi d'alléger la présentation de l'étude bibliographique du domaine et de présenter les éléments essentiels allant dans le sens de cette thèse.

Pour le lecteur intéressé uniquement par CELINE, le tableau ci-dessous permet d'accéder directement au spécifique.

Accès à CELINE		
Chapitre	**A lire**	**Contenu des paragraphes spécifiques**
Définition progressive d'un cahier de charge		
Chapitre 2 *Problématique, taxinomie et correction des erreurs*	A partir du § 2.3	1. Vocabulaire général adopté 2. Méthodes de correction retenues 3. Début de la détermination du tableau *structure de données* nécessaire pour satisfaire quelques choix fondamentaux : • prise de décision multi-niveaux, • coopération et mise en concurrence des différents modules (agents), • parallélisme des actions, • prise de décision différée.
Chapitre 3 *Les prototypes de correction des erreurs*	Rien de spécifique sur CELINE mais rejet de certaines solutions incompatibles avec les objectifs fondamentaux de CELINE et en sens inverse, repérage de solutions intéressantes et donc à mettre en œuvre avec éventuellement des adaptations.	
Chapitre 4 *Modèles humains en correction des erreurs*	Chapitre entier	1. Le système recherché incorpore dans sa prise de décision les spécificités du rédacteur. Il y a donc lieu de faire un apprentissage de ce rédacteur sur le plan des compétences, des domaines lexicaux utilisés etc. 2. CELINE accepte des agents imparfaits ou mal adaptés. Il faut donc leur affecter des *coefficients d'efficacité*, de *crédibilité* etc. 3. En vue d'une correction automatique, le système doit *s'auto-évaluer* lui-même.
Chapitre 5 *Les modèles de Markov*	Chapitre entier	Présentation des résultats du dépouillement de corpus et des matrices de transitions correspondantes
	A partir du § 5.5	La mise en œuvre et stratégies principales de désambiguïsation.

Chapitre 6 *Le modèle Linguistique Partiel Suffisant*	Chapitre entier	1. Ensembles des données spécifiant le rédacteur : • domaines et taux d'utilisation des lexiques, • tables lexicales, • tables de règles grammaticales, • taux d'erreurs • matrices de transitions, • règles contextuelles de correction. 2. Image virtuelle et Image locale d'un système de détection : correction des erreurs : Implémentation à deux niveaux.
Chapitre 7 *La méthode des structures*	Chapitre entier	Ce chapitre présente une méthode quantitative applicable pour la vérification correction des accords dont la finalité est une contribution aux levées de doute sur des hypothèses contradictoires
Chapitre 8 *Les systèmes multi-agents*	A partir du § 8.7	1. La prise en compte des exigences formulées pour le système attendu coïncident avec les définitions et objectifs des systèmes multi-agents. Un parallèle va donc mettre en place une modélisation multi-agents de CELINE. 2. La prise en compte du tableau *structure de données* va nous amener à choisir une communication par *tableau noir*. 3. La prise en compte du processus de coopération des agents va préciser cette communication par l'utilisation d'un *tableau noir parallèle* et nous pourrons alors modéliser le contrôle et la communication.
La réalisation		
Chapitre 9 *Le système CELINE*	Chapitre entier	Le chapitre précédent spécifiait CELINE comme système multi-agents. Ce chapitre va détailler certains points de la réalisation.

1.3.3 DÉTAIL PAR CHAPITRE

Une première partie du chapitre 2, ***Problématique, taxinomie des erreurs et méthode de correction,*** nous permettra de rappeler la problématique de l'analyse d'une langue naturelle en insistant sur les ambiguïtés et les problèmes d'encombrement mémoire si on ne veut pas perdre de solutions. Nous nous attarderons sur la méthode des CHARTS dont nous exploiterons par la suite les apports au niveau non-redondances des traitements liés à la structure de données utilisée.

Dans une deuxième partie, après une étude bibliographique de la taxinomie et de la classification des erreurs, nous examinerons plus spécialement les choix retenus par l'équipe TRILAN dans DECOR logiciel de détection et correction des erreurs lexicales (Cohard 88) et dans CORSYN prototype de traitement des erreurs au niveau syntaxique (Strube de Lima 90).

Pour terminer ce chapitre, en s'appuyant sur les deux premières parties, nous donnerons un premier aperçu sur quelques choix fondamentaux concernant le système CELINE. Les traitement choisis et la nécessité de collaboration multi-niveaux nous entraînera à découvrir notamment l'importante structure de données globale que doit gérer le système. Complétée aux fils des chapitres, cette structure de données va être la clé du fonctionnement de CELINE. Par ailleurs les besoins d'intervention sur cette structure nous amènera à l'architecture du système.

De la gestion de cette structure découlera une nouvelle conception d'un agent central traitement de texte, détecteur et correcteur d'erreurs sous la forme d'un SGBD travaillant en coopération avec d'autres agents experts. La phrase initiale et la phrase corrigée apparaîtraient alors comme de simples *vues* extraites des tables.

Le chapitre 3 ***Prototypes du domaine de la détection correcton d'erreurs*** va commencer par une synthèse rapide sur DECOR et CORSYN, réalisations dont les principes ont été déjà développés au chapitre 2.

Une autre réalisation ECLAIR (Letellier 93) fera l'objet d'une étude plus approfondie. Ce système se présente comme un système multi-experts et multi-lexiques d'analyse et de correction lexicales. Eclair est particulièrement intéressant car son cahier de charge, assez complet sur le plan du traitement orthographique, tient compte d'une possibilité de coopération avec un analyseur syntaxico-sémantique déterministe ANDI (Grandchamp 86), lui même intégré à CARAMEL un système multi-experts spécialisé dans le traitement automatique de la langue (nous le retrouverons au chapitre 8 avec les systèmes multi-agents).

Après les chapitres 2 et 3, présentant un aperçu du domaine de la correction d'erreurs, le chapitre 4 ***Modèles humains en correction des erreurs*** va nous rapprocher de l'adaptation au rédacteur.

Dans une première partie, après un exemple illustrant les besoins de connaissances sur le rédacteur, nous justifierons l'appellation de maître d'école par une comparaison, dans la démarche de connaissances de l'élève, du comportement attendu du système vis-à-vis du rédacteur et du comportement d'un maître d'école vis-à-vis de ses élèves.

Dans une seconde partie, face à des agents imparfaits, nous définirons conceptuellement des coefficients permettant de chiffrer la confiance du système dans les productions de ces agents. Le directeur du service de renseignements va prendre en charge cette mission.

Le modèle du maître d'école est en quelque sorte un modèle historique vécu. Sur le plan de la correction d'erreurs nos premiers souvenirs viennent de l'école primaire (avec exercices au tableau). C'est là que nous avons appris comment rechercher des fautes et les corriger et l'idée d'un système informatique reproduisant les mêmes démarches est séduisante. Aussi dans une troisième partie, nous définirons le protocole général de fonctionnement de CELINE en le calquant sur cette activité humaine scolaire : nous allons rechercher un système dans lequel des spécialistes coopèrent de la même manière que des élèves faisant des travaux de groupe et dans lequel les connaissances peuvent s'exprimer, non par des formalismes mathématiques sophistiqués abstraits et très loin de la réalité, mais par des règles analogues à celles rencontrées dans les manuels scolaires (exemple le Bescherelle).

Un apprentissage rigoureux du rédacteur demandant un soutien quantitatif, dans le chapitre 5 ***Les modèles de Markov*** nous présenterons dans une première partie les modèles de Markov ainsi que quelques algorithmes de mise en œuvre de filtres Markoviens.

Pour terminer, en se basant sur les résultats d'une expérimentation donnée en annexe, nous tirerons des conclusions pratiques d'emplois et nous déterminerons les règles retenues dans CELINE pour l'utilisation des modèles. Nous y verrons notamment comment CELINE peut changer de stratégie en fonction de la situation réelle et du rédacteur. Nous ne serons plus alors très loin de la mise en œuvre de la coopération entre agents.

Les exemples donnés dans ce chapitre seront tous relatifs au domaine de la désambiguïsation des catégories morphologiques. Mais nous avons déjà signalé que CELINE utilise également les modèles dans d'autres domaines comme par exemple la détermination de la première lettre d'une forme textuelle inconnue ou encore l'ordre des lettres à essayer.

Nous reviendrons ensuite au rédacteur et à sa définition par le chapitre 6 ***: Le modèle Linguistique Partiel Suffisant.***
Nous commencerons par nous intéresser à la reconnaissance lexicale, problème important compte tenu de la variété des domaines possibles.

Dans les fragments de protocole de travail sur la structure de données, nous avons décidé que dès les premiers mots reconnus, les analyseurs syntaxiques commençaient à rechercher les structures terminales. La validation d'une phrase finale va permettre de mémoriser les différentes structures composantes de l'arbre syntaxique. La deuxième partie du chapitre va donc s'intéresser à la détermination en extension de la grammaire du rédacteur.

En troisième partie, les modèles de Markov compléteront cette définition de la grammaire en permettant de calculer des poids pour les diverses règles.

Nos conclusions sur la non-convergence des matrices observée au chapitre V nous permettront alors d'introduire des corrections de ces modèles sous la forme de règles contextuelles de corrections.

Nous dresserons ensuite une liste récapitulative de l'ensemble des coefficients, seuils et taux divers par rédacteur.

En conclusion et pour terminer, cette définition du rédacteur, va nous permettre d'introduire une adaptation de l'architecture du système qui pour un utilisateur particulier va se simplifier en constituant *une image personnalisée du système de correction*.

Dans l'optique de corrections quantifiées, le chapitre 7 traitera de ***La méthode des structures***.
A partir des généralités dégagées, un paragraphe va tout d'abord présenter la notion de *structure* et de *qualité d'une structure*. Viendra ensuite un panorama des règles d'accord à implanter, dans lequel nous distinguerons des règles qualifiées de *règles d'unification* et des règles qualifiées de *règles d'anti-unification*. Cette distinction nous fera distinguer des *structures unifiables* basées sur des règles d'unification et des *structures absorbantes* basées sur des règles d'anti-unification. Pour terminer, nous examinerons une quantification du calcul de la qualité pour des structures terminales et des structures non terminales.

Les chapitres 2 à 7 ayant permis de préciser nos choix pour le système de correction des erreurs envisagé, le chapitre 8 ***Les systèmes multi-agents et l'IAD*** va permettre de préciser et justifier l'implémentation sous la forme d'un système multi-agents.

Une première partie va nous permettre de traiter successivement les points suivants :
- Comment l'utilisation conjointe du parallélisme et de l'intelligence artificielle permet de définir de nouveaux domaines et parmi ceux-ci le domaine des systèmes multi-agents.
- La notion d'agent et les regroupements d'agents : les sociétés. Nous dégagerons un ensemble de qualificatifs et de propriétés pouvant s'appliquer aux agents.
- Au sein d'une société, l'une des préoccupations consiste à coordonner les activités des agents d'où le besoin de communiquer. Cette communication généralement définie comme le comportement social de l'agent, peut prendre des formes extrêmement variables depuis la simple réponse à un stimuli d'un agent réactif jusqu'à une communication faisant intervenir les notions les plus fines du génie cognitif et aboutissant à des négociations entre agents. La notion de contrôle explicite la coopération entre les agents.

Une deuxième partie décrira des systèmes complets en se limitant au domaine du traitement des langues naturelles.

Suite logique, le chapitre 9 ***Le système CELINE*** présentera CELINE en tant que système multi-agents.
Dans un paragraphe d'introduction nous allons rappeler certains points généraux de la problématique justificatifs de certains compléments.

Dans une deuxième partie, nous présenterons des généralités sur l'architecture *hiérarchique pseudo-pyramidale hybride* du système, les modes de communications et pour terminer nous établirons une classification des agents.

Après des généralités sur le contexte de développement de CELINE et son paramétrage linguistique, une partie intermédiaire permettra de comprendre le passage d'un texte formaté à une structure de données représentative de ce texte. Une partie de cette structure permet l'initialisation du tableau noir.

Nous pourrons alors entreprendre la description du fonctionnement global par l'étude du contrôle au travers du fonctionnement du superviseur et des pilotes. S'intéressant au pilote des activités lexicales, nous découvrirons une méthode de recherche lexicale basée sur un *lexico-thésaurus*, méthode servant aussi bien à optimiser la recherche du bon lexique qu'à définir les domaines d'un utilisateur.

Une dernière partie récapitulera les agents développés ou en cours de développement et présentera une extension de la classification des agents lexicaux.

Pour terminer, le chapitre 10 ***Conclusion*** nous permettra, outre un bilan récapitulatif, d'envisager CELINE comme une réalisation encore futuriste misant sur une évolution possible des réseaux et de l'industrie des logiciels.

2. PROBLÉMATIQUE, TAXINOMIE DES ERREURS ET MÉTHODES DE CORRECTION

Finalité du chapitre

La taxinomie des erreurs a été un sujet de recherche pour les linguistes bien avant l'ère du traitement informatique et la littérature du domaine est depuis longtemps bien fournie.

En vue de la correction des erreurs, la connaissance des origines des erreurs est importante car elle permet souvent de déterminer des heuristiques de diagnostics et des heuristiques de corrections. Outre les règles générales valables quel que soit le rédacteur, cette connaissance contribue à la détermination du profil d'un rédacteur particulier.

Considérons par exemple la correction de la phrase *je manje une lomme* (exemple que nous approfondirons au § 2.3.5) :

- si nous savons que le rédacteur commet fréquemment des erreurs de transcriptions phonétiques, *manje* est corrigeable en *mange.*
- si nous savons que le rédacteur commet fréquemment des erreurs de frappe, *lomme* est corrigeable en *pomme* (*l* voisin du *p* sur le clavier).

En deux étapes nous obtenons la phrase corrigée *je mange une pomme*.

D'un point de vue informatique, le fonctionnement du système de correction est lié à la classification retenue.

Résumé du chapitre

La première partie nous permettra de rappeler la problématique de l'analyse d'une langue naturelle en insistant sur les ambiguïtés et les problèmes d'encombrement mémoire si on ne veut pas perdre de solutions. Nous nous attarderons sur la méthode des cartes dont nous exploiterons par la suite les apports au niveau non redondances des traitements liés à la structure de données utilisée.

Dans une deuxième partie, après une étude bibliographique de la taxinomie des erreurs, nous examinerons plus spécialement les choix retenus par l'équipe TRILAN dans DECOR logiciel de détection et correction des erreurs lexicales (Cohard 88) et dans CORSYN prototype de traitement des erreurs au niveau syntaxique (Strube de Lima 90).

Pour terminer en s'appuyant sur les deux premières parties, nous donnerons un premier aperçu sur quelques choix fondamentaux concernant le système CELINE. Les traitements choisis et la nécessité de collaboration multi-niveaux nous entraînera à découvrir notamment l'importante structure de données globale que doit gérer le système. Cette structure de données va être la clé du fonctionnement de CELINE et plusieurs chapitres nous permettront de la compléter. Par ailleurs les besoins d'intervention sur cette structure nous amèneront à l'architecture du système.

De la gestion de cette structure découlera une nouvelle conception d'un agent central traitement de texte, détecteur et correcteur d'erreurs sous la forme d'un SGBD travaillant en coopération avec d'autres agents experts. La phrase initiale et la phrase corrigée apparaîtront alors comme de simples *vues* extraites des tables.

2.1 LANGUES NATURELLES ET AMBIGUÏTÉS

2.1.1 PRÉSENTATION GLOBALE

Les langues naturelles possèdent de nombreuses caractéristiques qui sont à la base de l'efficacité du langage mais qui sont source de nombreux problèmes pour le traitement informatique (Winograd 72) (Sabah 88) (Sabah 89) (Amselem 91). Parmi les principales difficultés du traitement automatique des langues figurent les problèmes *d'ambiguïté.*

Il existe différents types d'ambiguïtés que nous pouvons différencier à travers quatre types de connaissances mises en jeu dans le processus d'interprétation : la morphologie, la syntaxe, la sémantique et la pragmatique. Les exemples suivants vont permettre de préciser ces différences.

Je prépare le mémoire

Cet énoncé semble limpide. Pourtant il existe effectivement une ambiguïté lexicale au niveau du mot *le* soit article défini soit pronom personnel. Notre cerveau, par la position des mots dans la phrase, en d'autres termes la syntaxe, détermine dans ce cas le sens unique de *le* et rend la deuxième interprétation impossible.

Je lis des livres

Cette fois, c'est notre savoir lié à la sémantique du verbe *lire* qui nous permet la levée de l'ambiguïté et nous fait choisir *livres* synonyme de *bouquins* et non *livres* unité de mesure de masse ou monnaie anglaise. Nous rejetons de même *livres* en tant que forme verbale dérivant de *livrer*

Il y a un os

Cette fois, malgré la simplicité de l'énoncé, l'interprétation définitive est impossible. L'ambiguïté réside au niveau du mot *os* qui peut vouloir dire *partie du squelette* ou encore *problème*. La levée d'ambiguïté nécessite donc dans ce cas des connaissances pragmatiques relatives au contexte de prononciation de cette phrase.

Sam a préparé le repas de Lyn

Même si le sens exact de chaque mot de cette phrase peut être déterminé, deux interprétations sont possibles selon le consommateur : *Sam a préparé le repas pour Lyn* ou *Sam a préparé le repas que Lyn avait apporté* soit pour lui-même soit pour une autre personne.

Ces quelques exemples donnent une faible idée de la complexité générale de la levée d'ambiguïté. Nous pourrions entre autres ajouter les problèmes liés aux ellipses, aux anaphores etc. ...

Parmi les différentes formes d'ambiguïtés, le premier exemple nous montre que certaines ambiguïtés lexicales peuvent être levées par la syntaxe. La réalisation d'un correcteur d’erreurs va donc devoir s’appuyer sur des analyseurs syntaxiques et cela demande donc de préciser cette notion d'ambiguïté lexicale.

2.1.2 COMPOSANTES LEXICALES DES AMBIGUÏTÉS

Précisons la notion d'ambiguïté lexicale à travers un exemple : *La belle ferme le voile*

Parmi les interprétations :

sens 1 :	La	jolie	ferme ("agricole")	le (?)	cache,
sens 2 :	La	personne "belle",	inflexible	le (?)	cache,
sens 3 :	La	personne "belle"	ferme	le	rideau,

Nous remarquons que les mots possèdent des catégories lexicales différentes.

	La	belle	ferme	le	voile
sens 1 :	déterminant	adjectif	substantif	pronom	verbe
sens 2 :	déterminant	substantif	adjectif	pronom	verbe
sens 3 :	déterminant	substantif	verbe	déterminant	substantif

La désambiguïsation va consister à préciser pour chaque mot la catégorie adéquate.

2.1.3 ANALYSES EN LANGUE NATURELLE

Nous avons vu que la levée d'ambiguïté pouvait concerner le mot d'une façon lexicale, ou bien l'interprétation d'un mot ou de la phrase par la syntaxe, la sémantique ou la pragmatique. D'une manière générale lorsque l'on désire traiter une phrase, on passe par différentes étapes qui, selon le codage des connaissances et/ou la stratégie ciblée, peuvent être simultanées ou complètement indépendantes.

L'analyse d'une phrase demande donc :

L'identification de chaque mot : analyse lexicale,

L'affectation pour chaque mot d'une liste des catégories possibles: analyse morphologique,

la reconnaissance des structures syntaxiques possibles : analyse syntaxique,

et à un niveau plus élevé

une connaissance sémantique des mots,

des connaissances pragmatiques.

En pratique, nous avons déjà compris lors du chapitre d'introduction que ces différentes analyses doivent être simultanées et complémentaires.

2.1.3.1 L'analyse lexico-morphologique

L'analyse lexico-morphologique essaye de reconnaître chacun des mots et fournit en cas de réussite la ou les catégorie(s) lexicale(s) et les variables associées.

Par exemple, l'analyseur morphologique de PILAF donne de la phrase :

j'appuie donc sur les touches correspondantes

la transduction suivante :

2/j'	pper	351	sin fem mas uno suj
7/appuie	verb	125	sin tre uno pre ind
7/appuie	verb	125	sin tre uno pre sub
7/appuie	verb	125	sin dos imp
5/donc	apdi	435	
5/donc	adv	109	
5/donc	coco	104	

4/sur	prep	105	
4/les	det	24	plu fem mas
4/les	pper	16	plu fem mas tre cod
8/touches	subc	10	plu fem
8/touches	verb	72	sin dos pre ind
8/touches	verb	72	sin dos pre sub
16/correspondantes	subc	287	plu fem
16/correspondantes	adjq	344	plu fem

Les renseignements donnés par cette analyse sont par exemple pour la première ligne :

j' pronom personnel
modèle de comportement grammatical : 351
nombre: singulier
genre : féminin ou masculin
personne : première
fonction : sujet

La rencontre de plusieurs lignes pour un même mot traduit le phénomène d'ambiguïté lexicale. Par exemple *touches* peut être un substantif commun ou un verbe.

2.1.3.2 La désambiguïsation des catégories morpho-syntaxiques

Selon la stratégie adoptée, la désambiguïsation lexicale élimine, en totalité ou en partie, les solutions multiples données par l'analyse morphologique. Cette élimination s'effectue par appel à l'utilisateur ou à l'aide de filtres.

On obtient :

2/j'	pper	351	sin fem mas uno suj
7/appuie	verb	125	sin tre uno pre ind
5/donc	adv	109	
4/sur	prep	105	
4/les	det	24	plu fem mas
8/touches	subc	10	plu fem
16/correspondantes	adjq	344	plu fem

Cette étape, très délicate, est souvent inexistante dans les réalisations classiques qui vont traiter partiellement les ambiguïtés.
En présence d'erreurs, le nombre d'ambiguïtés croît considérablement. Il semble donc important de traiter ce problème.

2.1.3.3 L'analyse syntaxique

L'analyse syntaxique**,** par application de règles grammaticales, donne la structure générale de la phrase et établit des liens entre les mots.

Par exemple pour la phrase *j'appuie donc sur les touches correspondantes*, en utilisant une grammaire de dépendances, on obtient l'arbre syntaxique suivant (figure 2.1) :

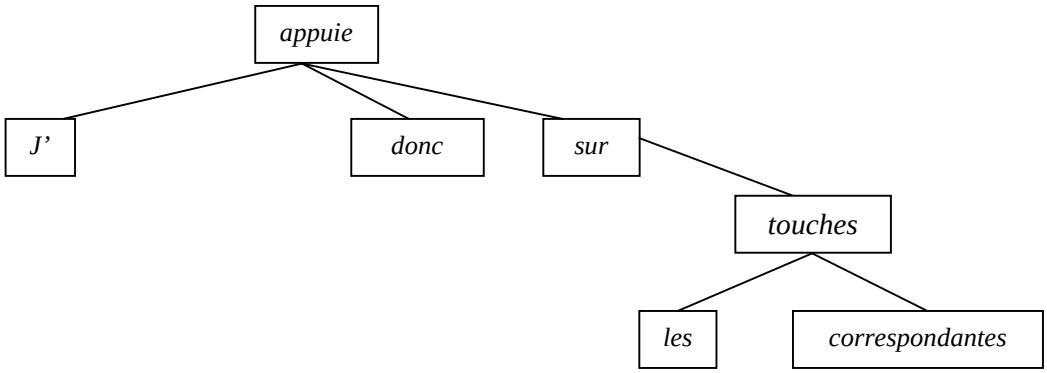


Figure 2.1

L'absence d'un étage de désambiguïsation est très pénalisant pour l'analyse syntaxique qui va devoir considérer un grand nombre de constructions intermédiaires. Dans le cas où cette analyse produit au moins un arbre complet, la pénalisation se traduit seulement par une complexité du travail. Dans le cas contraire, la forêt obtenue (liste d'arbres partiels) est difficilement exploitable.

Indépendamment des ambiguïtés et en cas d'erreur(s) sur la phrase (mots inconnus, inversion de mots, suppression d'un mot), la probabilité de ne pas obtenir d'arbre complet augmente fortement.

2.1.3.4 Les analyses sémantique et pragmatique

L'analyse sémantique établit également des liens entre les mots mais cette fois en se plaçant au niveau de l'idée qu'ils véhiculent.

Une possibilité consisterait à décorer un arbre syntaxique avec des renseignements dépendant du domaine du discours par exemple :

appuie APPUYER (Qui : ANIME
sur-quoi OBJET-PHYSIQUE)
touche OBJET PHYSIQUE (Usage : ORGANE-ENTREE
Méthode : APPUYER)

Suivant le domaine envisagé, différents sens plus ou moins probables peuvent être proposés. Par exemple pour *touche* il faudrait différencier la *touche d'un instrument* de la *touche de peinture*, de la *touche à la pêche*, *de la touche au football*, *touche sentimentale* etc.

L'analyse pragmatique tient compte du contexte dans lequel se situe la phrase et précise et affine l'analyse sémantique.

Les niveaux sémantique et pragmatique seront peu évoqués dans cette thèse. La diversité des modèles utilisés selon les domaines ne permet pas pour l'instant d'envisager une prise en compte systématique de ces niveaux dans un système robuste multi-domaines. Seuls des agents autonomes d'un domaine particulier (par exemple en météo ; voir § 9.2.4.4) permettront la prise en compte de ces niveaux pour les parties du discours relatives à ce domaine. Il y aura lieu de distinguer les performances du système CELINE en tant que système multi-agents de correction des erreurs et la profondeur d'analyse linguistique qui elle va être liée aux capacités des agents utilisés.

2.1.4 LANGUE NATURELLE ET GRAMMAIRE

Afin d'analyser automatiquement la structure syntaxique d'une phrase, il est indispensable de disposer de deux éléments :

- une grammaire qui est une description formelle des structures autorisées,
- une technique d'analyse de la phrase qui permet de déterminer sa structure selon les règles de la grammaire.

A la phrase :

Je possède une montre

correspond une structure syntaxique qui peut être représentée à travers des modélisations différentes :

Exemple 1 :

```
phrase  (
        sujet     (pronom personnel   (je))
        verbe     (posséder)
        complément d'objet direct   (groupe nominal (article indéfini  (une)
                                                                nom             (montre)))
        )
```

Cette écriture de la structure syntaxique correspond en réalité à un arbre écrit sous forme de liste parenthésée.

Exemple 2 avec une grammaire de constituants (figure 2.2) :

```
P       -->     SN      SV
SN      -->     PRO
SN      -->     DET     SUBC
SV      -->     VERB    SN
```

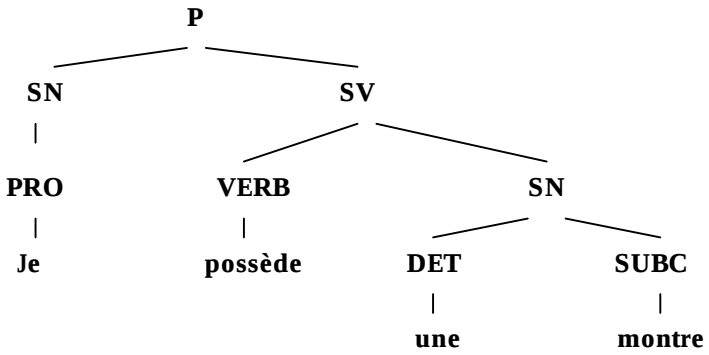


Figure 2.2

Exemple 3, avec une grammaire de dépendances (figure 2.3) :

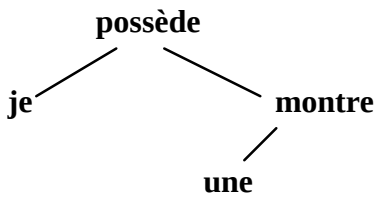


Figure 2.3

2.1.5 MÉCANISME D'ANALYSE

L'objet de ce paragraphe est :

1. D'éventuellement rappeler la complexité du phénomène d'analyse syntaxique en examinant un exemple simple avec une grammaire "complice".
2. Surtout de montrer que tous les modes d'analyse ne sont pas équivalents si on cherche à les mixer avec des vérifications multi-niveaux, comme par exemple, *vérifier à chaque règle acceptée que les variables morphologiques associées montrent qu'il n'y pas lieu de rejeter cette règle*, une autre solution consistant dans ce cas à *mettre en doute les formes textuelles*. Or une mise en doute de la forme textuelle et son remplacement par une autre forme textuelle casse le mécanisme de l'analyse qui doit complètement être reprise à zéro.

Soit à analyser la phrase

Le joli bateau bleu tangue brutalement

avec la grammaire (ou P est le symbole initial) :

(R1)	P	-->	NOM
(R2)	P	-->	VERB
(R3)	P	-->	GN GV
(R4)	GN	-->	DET NOM
(R5)	GN	-->	DET NOM ADJQ
(R6)	GN	-->	DET ADJQ NOM
(R7)	GN	-->	DET ADJQ NOM ADJQ
(R8)	GV	-->	VERB
(R9)	GV	-->	VERB ADV
(R10)	DET	-->	la ...
(R11)	DET	-->	le ...
(R12)	NOM	-->	Céline...
(R13)	NOM	-->	bateau...
(R14)	ADJQ	-->	joli
(R15)	ADJQ	-->	bleu
(R16)	VERB	-->	tangue
(R17)	VERB	-->	mange
(R18)	ADV	-->	brutalement

L'analyse de la phrase proposée conduit à l'arbre syntaxique de la figure 2.4 :

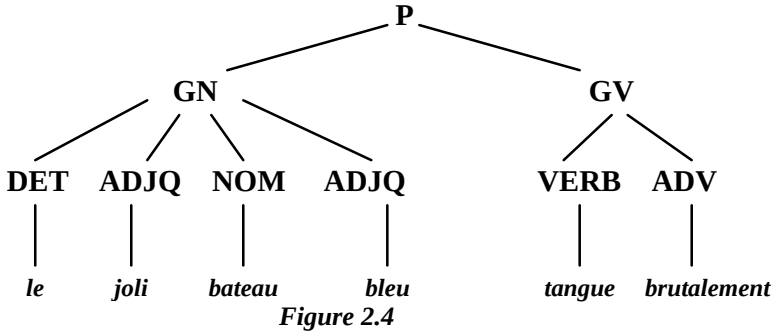


Figure 2.4

La méthode de reconnaissance utilisée correspond à un parcours préfixé de l'arbre syntaxique.

Une analyse syntaxique peut être conduite selon diverses stratégies. Nous allons utiliser en guise d'introduction une méthode qualifiée de "stratégie gauche droite en profondeur d'abord".

L'analyseur va successivement essayer d'appliquer R1, R2, R3. L'analyseur va procéder par essais successifs des diverses règles possibles. Si une règle n'est pas applicable l'analyseur va revenir en arrière et appliquer l'alternative suivante s'il en existe une. Nous présentons dans le tableau de la page suivante les essais successifs correspondants à cette analyse.

L'analyse du groupe nominal de l'exemple ci-dessus (voir tableau page suivante), nous montre la complexité et la redondance des recherches lors d'une analyse syntaxique grammaticale. Or nous voulons, de plus, mener en parallèle les autres niveaux d'analyses et faire intervenir par exemple les variables morphologiques. La forme d'analyse présentée se prête mal à cette interpénétration des niveaux. Il faudrait par exemple à chaque règle non rejetée et acceptable, examiner ce qu'il en est des variables morphologiques et des accords avant de conclure à son acceptation. Dans le cas d'un désaccord, il faudrait alors rejeter la règle et revenir au point de choix précédent.

Même en admettant une phrase sans erreurs, nous voyons, dans le tableau des analyses, que nous revenons plusieurs fois sur la reconnaissance des mêmes mots. Il faudrait donc reprendre aussi les vérifications multi-niveaux correspondantes ce qui alourdirait encore plus le processus.

Nous voyons donc se profiler trois exigences :

- Il faut trouver un mécanisme différent d'analyse permettant cette interpénétration des niveaux.
- Il faut mémoriser les étapes intermédiaires pour éviter la redondance des essais.
- La progression sur la phrase doit permettre une modification de celle-ci (par exemple ajout de formes textuelles concurrentes sans casser la totalité du mécanisme d'analyse.

La méthode des cartes va nous permettre de satisfaire « facilement » nos exigences et nous allons donc, comme pour l'analyse descendante précédente, suivre une partie d'une analyse afin de mettre en évidence cette possibilité.

L'analyseur "essaie" **R1** **Point de choix 1 entre R12 et R13** **Point de choix 1 entre (R12 rejetée) et R13** **règle R1 rejetée**	L'analyseur "essaie" R12 : Echec **Retour au point de choix 1** L'analyseur "essaie" R13 : Echec **Retour au point de choix 1**	
L'analyseur "essaie" **R2** **Point de choix 2 entre R16 et R17** **Point de choix 2 entre (R16 rejetée) et R17** **règle R2 rejetée** L'analyseur "essaie" R3 **Point de choix 3 entre R4, R5, R6, R7**	L'analyseur "essaie" R16 : Echec **Retour au point de choix 2** L'analyseur "essaie" R17 : Echec **Retour au point de choix 2** L'analyseur "essaie" **R4** **Point de choix 4 entre R10 et R11** **Point de choix 4 entre (R10 rejetée) et R11** **Point de choix 5 entre R12 et R13** **Point de choix 5 entre (R12 rejetée) et R13** **Règle R4 rejetée** **Retour au point de choix 3**	L'analyseur "essaie" R10 : Echec **Retour au point de choix 4** L'analyseur "essaie" R11 : OK ***"le"* est reconnu** L'analyseur "essaie" R12 : Echec **Retour au point de choix 5** L'analyseur "essaie" R13 : Echec **Retour au point de choix 5**
Point de choix 3 entre (R4 rejetée), R5, R6, R7	*Nous allons trouver ici exactement la même recherche que dans l'essai de R4 précédent puisque la règle R5, comme la règle R4, commence par la reconnaissance de la séquence DET NOM.*	
Point de choix 3 entre (R4, R5 rejetées), R6, R7	*Nous allons refaire encore la reconnaissance du "le" (3 ième fois) Puis reconnaître "joli" comme adjectif et "bateau" comme nom avant de rejeter la règle.*	
Point de choix 3 entre (R4, R5, R6 rejetées), R7	*Nous allons refaire la reconnaissance du "le" (4 ième fois), de l'adjectif "joli" (2 ième fois), du nom "bateau" (2 ième fois), puis la reconnaissance de l'adjectif "bleu",avant de valider la règle*	

2.1.6 MÉTHODE DES CARTES

2.1.6.1 Introduction à la méthode

Nous remarquerons tout d'abord son nom complet prometteur *: méthode des tables des sous-chaines bien formées*. Qui dit sous-chaînes dit partie de la phrase et depuis le chapitre 1 d'introduction, nous souhaitons travailler sur des fragments de phrases.

Lors d'une analyse classique ascendante ou descendante, lorsqu'un choix s'impose, on est confronté au problème du non-déterminisme. La gestion du choix peut être réalisée de deux façons différentes :

1. On évalue toutes les solutions en parallèle.
2. On évalue séquentiellement les choix avec mémorisation des points de choix et ensuite retour arrière.

Les différentes stratégies d'analyse ont chacune des avantages et des inconvénients :

1. La stratégie descendante évite d'envisager des combinaisons qui n'aboutiraient pas à la structure complète de la phrase. Par contre, l'utilisation du retour arrière duplique beaucoup des analyses déjà effectuées lorsque l'on revient sur la même sous-chaîne à travers des chemins distincts.
2. La stratégie d'analyse en parallèle rend possible la combinaison des sous-calculs communs mais elle entraîne aussi beaucoup de calculs inutiles.

Méthode	Inconvénient(s)	Avantage(s)
Evaluation en parallèle (largeur d'abord)	Très coûteux en terme d'encombrement mémoire car on doit mémoriser toutes les solutions possibles.	En choisissant la méthode de programmation, on arrive à "factoriser" des sous-calculs communs.
Evaluation séquentielle et retour arrière (profondeur d'abord)	On travaille en aveugle sans guide et la mémoire de tout ce que l'on a fait avant est perdue. On ne peut pas "factoriser" les calculs et on va faire des calculs répétitifs.	On construit une seule solution à la fois ce qui donc est peu coûteux du point de vue encombrement mémoire.

Nous observons donc que les deux méthodes possibles sont opposées dans leurs avantages et leurs inconvénients. Pour pallier ce problème, diverses méthodes ont été imaginées. Parmi elles, la méthode des cartes (KAY 64) repose sur une méthode de programmation dynamique par une mémorisation des résultats partiels de manière à éviter les redondances des calculs.

2.1.6.2 Principe de la méthode des cartes

2.1.6.2.1 Inspiration de la méthode des cartes

L'idée clé inspiratrice de la méthode des cartes est de garder la trace des analyses déjà effectuées c'est-à-dire conserver l'interprétation grammaticale de sous-chaînes, interprétation pouvant être réutilisée par d'autres règles de la grammaire.

Un problème réside dans la représentation des structures partielles identifiées qui peuvent faire partie ou non de l'analyse finale complète.

Par exemple, dans la phrase *Le lapin mange des carottes crues*, le fait d'affecter à *crues* la catégorie *adjectif* dans une partie de l'arbre de recherche ne veut pas dire obligatoirement qu'il restera un adjectif dans l'analyse complète. Dans cet exemple, *adjectif* restera la catégorie attribué à l'issue de l'analyse mais dans d'autres phrases, la catégorie finale retenue pourrait être un verbe ou un substantif commun par exemple.

Il est donc nécessaire de pouvoir mémoriser les différentes alternatives des analyses partielles de la chaîne, sans s'engager à prendre une décision prématurée sur ce qui est correct.

Autre problème : selon l'ensemble des règles utilisées, la grammaire G peut donner plusieurs arbres syntaxiques pour la phrase P. Cela provient principalement des ambiguïtés morphologiques (*la belle ferme le voile)* ainsi que de possibilités de dépendances multiples entre les mots (*La maison de la tante que j'ai vue ...)* avec comme question : *a-t-il vu la tante ou la maison de la tante ?*). L'analyse va alors aboutir à une forêt d'arbres complets pour la même chaîne. La levée de ces ambiguïtés syntaxiques demanderait une analyse sémantique ou pragmatique de la phrase et des structures trouvées.

Il faudrait une structure de travail permettant de garder la trace des analyses effectuées séparément pour chaque constituant, on pourrait alors définir un algorithme qui combine les avantages de toutes les stratégies et permettrait par ailleurs le choix de la règle à appliquer à un moment donné. C'est ce que va concrétiser la méthode des cartes.

Pour comprendre la mise en œuvre de cette méthode nous allons successivement étudier les rubriques suivantes :

- définition précise de la carte des chaînes bien formées,
- conventions complémentaires et notion de carte active,
- règle fondamentale de travail,
- initialisation de la carte,
- stratégie d'invocation des règles,
- algorithme général d'analyse,
- stratégie de recherche.

2.1.6.3 Définition d'une carte

2.1.6.3.1 Présentation d'une carte ou TSBF

Une carte ou Table des Sous-chaînes Bien Formées est un graphe où :

- Les sommets représentent une position dans la chaîne à analyser : l'intervalle entre deux symboles consécutifs de cette chaîne.
- Les arêtes ou arcs sont étiquetés par un symbole de la grammaire.

Dans une carte, les indices des sommets de début et de fin de la chaîne sont numérotés 1 et n + 1 respectivement.
Une arête du graphe sera désignée par deux indices : celui du sommet de départ de cette arête et celui du sommet d'arrivée de cette arête. L'arête est décorée par un symbole de la grammaire.

Structure d'une phrase analysée lexico-morphologiquement et début de définition du graphe											
Soit n le nombre de mots de la phrase. Les mots sont repérés par un indice m tel que (1 < = m < = n):											
		mot[1]		mot[2]		. . .		mot[k]	. . .		mot[n]
		{		{		. . .		{	. . .		{
Ambiguïté Lexicale par mot		cat1 cat2 cat3 cat4 cat5		cat6 cat7 cat8		. . .		cat_u	. . .		cat9 cat10 cat11
		}		}		. . .		}			}
Sommets du graphe :											
	1		2		3	. . .					
Soit i l'indice désignant un sommet du graphe c'est-à-dire un intervalle entre deux mots (1 < = i < = n + 1)											

Nous nous inspirerons de ce graphe pour définir au paragraphe 2.3.5 la structure de données nécessaire à CELINE.

La CARTE donne pour une paire de sommets i, j (1 <= i < j <= n+1), quelles sont les catégories associables à la sous-chaîne des mots trouvés entre i et j.

La CARTE est un graphe orienté (chaque arc a une direction particulière), ne comportant pas de circuit, avec un seul nœud de départ et un seul nœud de fin.

2.1.6.3.2 Carte et relations d'ordre

La structure choisie permet à la CARTE d'ordonner directement la phrase :

> Un constituant avec un arc allant du nœud i au nœud j dans la table précède un constituant avec un arc de m à n si j < m.

Cette relation d'ordre ne se retrouve pas dans les structures d'arbres, ces derniers ordonnant directement à travers les constituants frères seulement. Par contre, avec une structure d'arbres, les relations de dominance immédiate indiquent quelles phrases font partie d'autres phrases.

Pour retrouver les constructions syntaxiques sous forme d'arbres, on peut étiqueter chaque arête par le symbole correct mais également par sa décomposition grammaticale correcte.

2.1.6.3.3 Visualisation d'une carte

Une carte peut être visualisée sous la forme d'un graphe (figure 2.5) où :

- les *sommets* (les noeuds) représentent une position dans la chaîne à analyser (l'intervalle entre deux symboles successifs de cette chaîne),
- les *arêtes* (les arcs) sont étiquetés par un symbole de la grammaire.

Au départ, seules les arêtes liant deux sommets adjacents (les mots individuels) apparaissent dans la carte, chaque arête portant une des catégories lexico-syntaxiques du mot correspondant (Det, Nom, Verb, etc).

Exemple sur la phrase *le feu menace* :

On a fait apparaître ici (figure 2.5) l'ambiguité lexicale du mot "le" qui est à la fois pronom et déterminant, ainsi que l'ambiguité de "menace", à la fois verbe et substantif commun.

Ensuite, après une analyse partielle, on obtiendra :

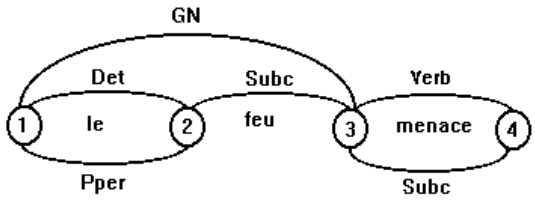


Figure 2.5

L'arête GN liant les sommets 1 et 3 indique qu'un groupe nominal a été reconnu sur la portion de phrase comprise entre les 2 sommets (figure 2.5).

Pour permettre la construction des arbres syntaxiques, les arêtes doivent porter, en plus d'un symbole de la grammaire (GN, GV, S,...), la décomposition du symbole (GN -> Det Subc,...) qui a permis de la créer (figure 2.6).

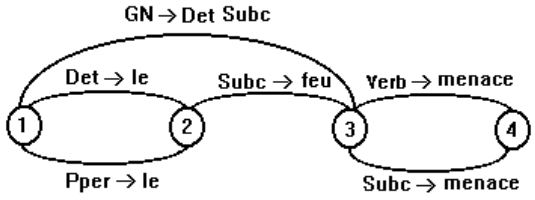


Figure 2.6

Ce qui correspond au sous-arbre (figure 2.7):

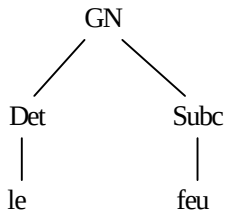


Figure 2.7

Il peut être intéressant de mémoriser d'autres informations que celles stockées sur la carte. Si une règle permet de reconnaître un élément dans une analyse partielle, alors il faudrait garder la trace des règles qui ont été essayées de celles qui ne l'ont pas été.

2.1.6.3.4 Une carte active

La carte décrite ci-dessus, sauvegarde le travail et évite que les sous-chaînes trouvées avec succès soient analysées plusieurs fois. Cependant la carte garde seulement la trace des analyses effectuées mais non la trace des hypothèses qui ont auparavant échoué.

De plus, les calculs dans différentes parties de l'espace de recherche sont exécutés pour des groupes en cours d'analyse, et rien n'empêche que le travail soit fait deux fois avant que l'analyse aboutisse. La seule façon de factoriser les sous-calculs communs et d'éviter la redondance d'essais déjà effectués est d'avoir une représentation explicite pour les buts et les hypothèses que l'analyseur fait à tout moment.

La carte est un bon outil de représentation des réalités structurelles, mais comment peut-on représenter les hypothèses et les buts ?
Pour satisfaire ce besoin, il est nécessaire d'amener deux améliorations à la structure de données de la TSBF telle que nous l'avons définie jusque là :

a) Au lieu d'imposer un graphe orienté strictement acyclique, nous allons donner un peu de souplesse en permettant des boucles (arcs où le nœud de début et le nœud de fin coïncident) mais l'interdiction générale de tout autre type de cycle reste valable (figure 2.8).

Figure 2.8

b) L'étiquette sur les arcs va pouvoir consister en une simple catégorie mais également en une règle décorée de la grammaire (figure 2.9).
Pour ce faire, on notera une arête avec la règle qu'elle porte à laquelle on ajoute:

- à gauche le numéro du sommet de départ et
- après le dernier symbole reconnu de la partie droite, le numéro du sommet d'arrivée (notation de Winograd 83).

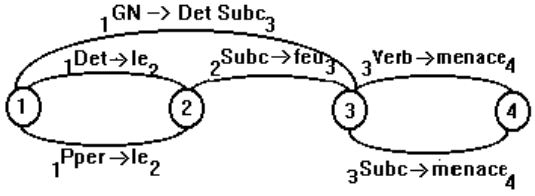


Figure 2.9

Récapitulons : la structure d'une arête dans une carte est donc caractérisée par les données :

- nœud de début
- nœud d'arrivée,
- le libellé : symbole en partie gauche de la règle de la grammaire que porte l'arête (qui sera toujours un élément du vocabulaire non terminal de la grammaire).
- liste correspondant aux termes déjà reconnus de la partie droite de la règle portée par l'arête,

- liste correspondant aux termes restant à reconnaître de la partie droite de cette règle.

Exemple : pour l'arête **$_i$S --> GN$_j$ GV**, on désigne par :

nœud de début :	i
nœud de fin :	j
libellé :	S
partie confirmée :	GN
partie à confirmer :	GV

On appellera *carte active* la carte lorsqu'on la complète avec les conventions a et b définies ci-dessus**.**

2.1.6.3.5 Arêtes partielles et arêtes complètes

Au cours de l'analyse, on distingue les arêtes partielles et les arêtes complètes :

- Une arête est *partielle* si on n'a pas encore complètement reconnu la partie droite de la règle qu'elle porte; c'est-à-dire si la partie restant à confirmer n'est pas vide.
- Sinon elle est *complète*.

Remarque : compte tenu de la façon dont on initialise la carte, une arête portant un symbole terminal est toujours complète (ex: $_1$Det-->le$_2$).

Exemple :

On a dans cette carte (figure 2.10), les trois arêtes :

A_1 : $_1$GN-->$_1$Det Subc
A_2 : $_1$GN-->Det$_2$ Subc
A_3 : $_1$GN-->Det Subc$_3$

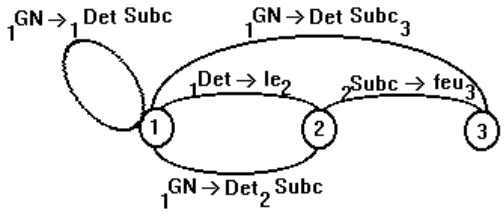


Figure 2.10

La première et la deuxième signifient que la reconnaissance du GN n'est pas terminée. L'arête A_1 part du nœud 1 et arrive à ce même nœud, alors que A_2 part du nœud 1 et arrive au nœud 2. Ces deux arêtes sont partielles.
La troisième arête est complète, elle part du nœud 1 et arrive au nœud 3 figurant à l'extrême droite de cette règle.

- Les arêtes partielles (type A_1 et A_2) correspondent à des hypothèses non confirmées complètement, elles sont dites *arêtes actives.*
- Les arêtes complètes (type A_3) correspondent à des hypothèses déjà confirmées et sont dites *arêtes inactives*.

2.1.6.3.6 Utilisation d'une carte dans les analyseurs

La carte active tout comme la TSBF, peuvent être utilisées pour des analyses selon diverses stratégies correspondant aux diverses combinaisons des options :

- analyse ascendante/descendante,
- analyse gauche/droite,
- analyse en profondeur d'abord/largeur d'abord.

Ces deux structures de données (TSBF et carte active) sont complètement neutres vis-à-vis de toutes les stratégies d'analyse. L'utilisation de ces structures permet simplement d'éviter à l'analyseur de recommencer des analyses déjà faites.

Tous les analyseurs exploitant une TSBF ou une carte comme structure de données sont qualifiés "d'analyseurs de carte" ("Chart parser").
La méthode des cartes a été et est encore très largement utilisée dans les systèmes de TALN comportant un noyau hors contexte (Earley 70) (Kaplan 73) (Lallich-Boidin 86).

2.2 TAXINOMIE DES ERREURS

2.2.1 LE NIVEAU LEXICAL

2.2.1.1 Typologie des erreurs lexicales

A travers quelques exemples, ce paragraphe va nous permettre d'acquérir un vocabulaire de base permettant une certaine technicité dans la désignation des erreurs. Il existe presque autant de classifications que de systèmes de correction d'erreurs mais les distinctions essentielles sont présentées ci-dessous.

Il est bien évident que face à une erreur isolée, l'interprétation de cette erreur pourra être multiple. Par exemple le deuxième t de *batteau* vient-il d'une erreur typographique de frappe sur le clavier ou d'une méconnaissance de l'orthographe ? Au point de vue de la correction, la classification d'une erreur isolée importe peu. Face à un ensemble d'erreurs, par contre, une statistique sur l'utilisateur pourrait rendre plus probable une erreur d'un type par rapport à une erreur d'un autre type et il y a là matière à réflexion car cela correspond à une possibilité efficace de choix dynamique d'une stratégie de correction face à un utilisateur particulier.

2.2.1.1.1 La classification de Catach, Duprez et Legris

Cette classification d'erreurs (Catach 80 b) est induite par le problème de l'enseignement de l'orthographe de la langue française. Nous allons résumé leur classification dans le tableau ci dessous :

Classes d'erreurs	***Remarques***	***Origines possibles des fautes***	***Exemples***
Erreurs à dominante phonétique (0rigine possible : mauvaise perception du son)		Omission ou adjonction d'une lettre ou d'une syllabe	*maitenant / maintenant* *arbruste / arbuste* *bienteur / bienfaiteur*

Classes d'erreurs	***Remarques***	***Origines possibles des fautes***	***Exemples***
Erreurs à dominante phonogrammique	Altérant la valeur phonique	Omission ou adjonction	*boef / boeuf* *cheuveu / cheveu* *exès / excès*
		Confusion	*oisis / oasis* *nè / né*
		Inversion	*idoit / idiot* *ceuillir / cueillir*
(Origine possible : perception du son mais méconnaissance de la transcription)	N'altérant pas la valeur phonique	Omission ou adjonction	*sin / sein* *tiket / ticket* *pensser / penser*
		Confusion	*invantère / inventaire* *pharmatie / pharmacie*
Erreurs à dominante morpho grammique	morphèmes grammaticaux	relations mal établies entre catégories grammaticales	*des ombres passes*
		entre nom et déterminant	*la routes; les rue*
		entre nom ou adjectif et complément	*sac de bille*
		formes du pluriel	*chevaus / chevaux*
	morphèmes lexicaux	non reconnaissance des mots	*névrie*
		Ignorance : de la famille lexicale	*inabité*
		des préfixes ou suffixes	*anterrement*
		du maintien ou non du radical	*nous vogons*
		des lettres finales justifiables	*couvers / couverts* *blan / blanc*
Erreurs concernant les homophones (Origine possible : mauvaise reconnaissance du mot)		de discours	*l'arme / larme* *encore sage / en corsage*
		lexicaux	*chant / champ* *voie / voix* *vain / vin*
		grammaticaux	*où / ou* *sont / son* *ces / ses*
Erreurs concernant les idéogrammes		majuscule, apostrophe, ponctuation etc ...	*l'état / l'Etat*
Erreurs concernant les lettres non fonctionnelles		consonnes doubles non fonctionnelles, les lettres latines et grecques h muet, e muet morphème du féminin...	

2.2.1.1.2 La classification de Véronis (Véronis 88)

Cette classification repose sur les notions de *compétence* et de *performance* :

- les erreurs de compétence reposent sur une mauvaise connaissance de la langue ;
- les erreurs de performance reposent sur un emploi effectif de la langue compte tenu du contexte.

Remarquons que Véronis introduit une catégorie de pseudo-fautes, qui ne sont pas celles du texte, mais celles liées aux insuffisances de la machine.

L'utilisateur et le système (ensemble machine plus logiciel) interviennent en tant qu'interlocuteurs, ce qui introduit quatre classes d'erreurs :

- erreurs de compétence de l'utilisateur (par exemple *hortodoxe / orthodoxe*),
- erreurs de performance de l'utilisateur (par exemple, inversion de deux lettres lors de la frappe : *segrent / sergent*),
- erreurs de compétence du système (par exemple mot inconnu dans le lexique),
- erreurs de performance du système (défaillance du système ou insuffisance d'un algorithme).

Pour terminer, signalons que Véronis divise ces grandes classes d'erreurs par niveau (lexical, syntaxique, sémantique) ce qui subdivise les erreurs en 12 groupes.

La stratégie de correction est différente selon le groupe et par exemple au niveau lexical, la stratégie de correction des erreurs phonographiques tient compte des erreurs typographiques et des erreurs phonétiques.

2.2.1.1.3 La classification des erreurs issus de VORTEX (Pérennou 90)

Cette classification introduit les opérations mentales, neuromotrices, mécaniques et/ou électroniques indispensables pour aboutir à la création d'un texte à corriger.

Cette création peut se diviser en deux phases :

1. Une phase conceptuelle pendant laquelle le rédacteur, partant des idées (*contenu conceptuel*) qu'il veut exprimer, construit ce qu'il va dire (*texte mental*). Durant cette phase, le rédacteur peut engendrer des *fautes linguistiques.*
2. Une phase de transcription du texte mental en *texte électronique* pendant laquelle vont intervenir les interfaces entre le rédacteur et le système informatique (clavier, lecture optique de caractères etc.). Les fautes aléatoires introduites seront appelées *fautes typographiques et* fautes d'édition.

Dans la réalité, les deux phases peuvent être simultanées (par exemple lors de la saisie directe d'un texte par un rédacteur).

Fautes linguistiques	Fautes de styles Fautes syntaxiques Fautes d'accord Fautes d'usage dont : • barbarismes, • fautes phonographiques
Fautes d'édition Fautes typographiques	Fautes d'éditions (par exemple : mauvais découpage en fin de ligne, mauvaise disposition des lignes etc.). Fautes typographiques : • effacement d'une lettre • insertion d'une lettre dont le doublement • substitution d'une lettre par une autre • transposition de deux lettres consécutives • coupure d'un mot • soudure de deux mots consécutifs

Classement des fautes en fautes linguistiques, d'édition et typographiques

Le système VORTEX (Pérennou 86) corrige les fautes d'usage et les fautes typographiques.

2.2.1.1.4 La classification des erreurs dans les systèmes DECOR et CORSYN

2.2.1.1.4.1 Les erreurs typographiques

Ces erreurs regroupent en réalité les erreurs désignées communément par fautes d'orthographe et les erreurs purement typographiques.

Exemples d'erreurs typographiques :

- les erreurs d'*omission* (Exemple : *solire / solaire*);
- les erreurs d'*insertion* (Exemple : *bâtioment / bâtiment*),
- les erreurs de *substitutions* (Exemple : *avzncer / avancer*),
- les erreurs de *transposition* : (Echange de deux lettres voisines; exemple : *particuleir / particulier*)
- les erreurs d'*accentuation*, les *coupures*, les *liaisons anormales* :
  accent aigu : *ecraser / écraser*,
  accent grave : *siecle / siècle*,
  accent circonflexe : *chaine / chaîne*,
  coupure : *par fois / parfois; s'en paler / s'empaler*
  liaisons anormales : *s'enaller / s'en aller*.

Exemples d'erreurs orthographiques :

- les erreurs d'*omission* (Exemple : assetion */ assertion*);
- les erreurs d'*insertion* (Exemple : lidvre */ livre*),
- les erreurs de *substitutions* (Exemple : *chirn / chien*),
- les erreurs de *transposition* : (Exemple : *aréoport* / aéroport.

2.2.1.1.4.2 Les erreurs phonétiques

Cette classe d'erreurs traduit la transcription oral-écrit c'est-à-dire la correspondance entre la prononciation et l'orthographe du mot. Une erreur de ce type correspond souvent à une erreur de compétence. On utilise ce mot à l'oral, on écrit une chaîne qui se prononce comme ce mot mais on ne connaît pas son orthographe.

Exemples d'erreurs :

- *assersion / assertion*
- *okurance / occurrence*
- *ortografe / orthographe*
- *vinku / vaincu.*

2.2.1.1.4.3 Les erreurs de génération

Ces erreurs, à classer également dans les erreurs de compétence, sont souvent commises en phase d'apprentissage de la langue (enfant, utilisation d'une langue étrangère). Elles correspondent souvent à une méconnaissance des règles grammaticales (règles de formation du pluriel, formes irrégulières verbales etc.)

Exemples d'erreurs :

- *allerons / irons*
- *chevals / chevaux*
- *chacaux / chacals*

2.2.1.2 Méthodes de correction

2.2.1.2.1 Méthodes locales et globales

Fouqueré (Fouqueré 88) distingue deux types de méthodes de correction lexicale :

Les *méthodes locales* ou *méthodes statistiques* caractérisées par des règles statistiques sur la construction des mots. La correction des mots consiste en un parcours sur un réseau de transitions valué.

Les *méthodes globales* ou *méthodes combinatoires* qui demandent de définir un moyen de comparaison entre deux chaînes de caractères. L'écart entre les deux chaînes est évalué grâce à une *fonction de comparaison***.**

2.2.1.2.2 La méthode de Damereau

En analysant la nature des erreurs lexicales des textes, Damerau (Damereau 64) a constaté que 80% des mots erronés ne comportent qu'une faute et que cette faute est de 4 natures différentes :

effacement d'une lettre (D),
insertion d'une lettre (I),
substitution d'une lettre par une autre (S),
transposition de deux lettres adjacentes (T).

Méthode :

Le recherche du mot correct va être faite dans le dictionnaire en ne conservant comme mot corrigé possible que ceux satisfaisant la double condition :

1) longueur du mot correct = longueur du mot incorrect à plus ou moins un caractère.
2) pas plus d'une différence sur les caractères composant le mot représentant une solution potentielle du mot incorrect.

Trois algorithmes sont alors possibles et choisis suivant que la longueur de la solution potentielle est inférieure, égale ou supérieure à celle du mot à corriger.

Cette méthode de correction permet de corriger 95 % des erreurs d'insertion, d'omission, de substitution ou de transposition soit 80%*95% = 76 % du total des erreurs lexicales.

2.2.1.2.3 La distance de Levenshtein

La distance de Levenshtein (Levenshtein 66) entre deux chaînes exprime le coût minimum pour passer de l'une à l'autre en n'utilisant que des opérations d'insertion, d'omission, de substitution ou de transposition.

De nombreux variantes de cette distance ont servi de supports à des développements pratiques. La notion de distance est fréquemment utilisée en technique de reconnaissance de formes, par exemple en milieu bruité.

2.2.1.2.4 Wagner et Fischer

Wagner et Fischer (Wagner 74) ont défini un algorithme de calcul de la distance en ne considérant que les opérations d'insertion, d'omission ou de substitution de caractères.

Notations :

aXq représente la substitution de a par q (*chqîne / chaîne*),
∑+r représente l'insertion de la lettre r *(livrre / livre*),
∑-r représente l'omission de la lettre r (*vere / verre*).

Exemple :*eaukurense* peut être considérée comme transformée en *occurrence* par la suite d'opérations : oXe, ∑+a, ∑+u, kXc, ∑-c, ∑-r, cXs

Algorithme de calcul

Soient les deux chaînes $x = x_1,...,x_n$ et $y = y_1,...,y_p$

Supposons qu'une différence existe entre la lettre d'indice i de la première chaîne et la lettre d'indice j de la deuxième chaîne.

```
Calcul_distance (x, y);
    fonction distance(i, j);
    début
        si x_i=λ et y_i=λ alors distance = 0
        sinon
        distance = minimum
        {distance(i-1, j-1)+C(x_i, y_j) ;      on remplace x_i par y_j
        distance(i-1, j)+C(x_i, λ),            on supprime x_i
        distance(i, j-1)+C(λ, y_j)}            on supprime y_j
    fin

    début
        distance(n,k);
    fin
```

avec C représentant une fonction de coût :

$C(x_i,y_j)$: coût de la substitution de x_i par y_j,

$C(\lambda, y_j)$: coût d'insertion de y_j,

$C(x_i, \lambda)$: coût d'omission de x_i.

Ces coûts peuvent être déterminés statistiquement.

Lowrance et Wagner

(Lowrance 75) Extension permettant de prendre en compte les transpositions entre deux caractères consécutifs.

Owalabi, McGregor puis **Okuda, Tanaka, Kasai** (Strube de Lima 84)
Introduction de variantes permettant de faire face aux problèmes de coût de traitement des algorithmes précédents.

2.2.1.2.5 Les clés de similarité

Le principe général (Cohard 88) (Strube de Lima 90) consiste à déduire une chaîne de caractères, ***la clé***, à partir du mot erroné. La consultation d'un dictionnaire de clés permet ensuite de retrouver le mot exact ou un ensemble de solutions concurrentes. Il existe différentes méthodes de construction d'une clé.

Définition de la clé dite **clé squelette** proposée par Pollock et Zamora (Pollock 84) :

Cette clé est obtenue par concaténation de
1) La première lettre du mot.
2) Les consonnes (une seule occurrence) par ordre d'apparition dans le mot.
3) Les voyelles (une seule occurrence) par ordre d'apparition dans le mot.

Exemples de clés squelettes :

Chaîne	Clé	Type d'erreur
OCCASION	OCSNAIO	
OCASION	OCSNAIO	Effacement d'un C
OCCAZION	OCZNAIO	Substitution de S par Z
INONDER	INDROE	
INNONDER	INDROE	Insertion d'un N
AÉROPORT	ARPTEO	
AÉROPOR	ARPEO	Effacement du T
ARÉOPORT	ARPTEO	Transposition du R et du E
AEEROPORT	ARPRTEO	Insertion du E
AEORPORT	ARPTEO	Transposition entre O et R
AEROPOFT	ARPFTEO	Substitution d'un R par F
AIROPORT	ARPTIO	Substitution de E par I

Nous voyons que le mot aéroport et les formes fausses dérivées ont, soit la même clé, soit des clés voisines (à une lettre près).

Pollock et Zamora justifient cette clé avec les observations suivantes :
1) La première lettre est la plus rarement mal orthographiée.
2) Les consonnes portent plus d'informations que les voyelles.
3) L'ordre des consonnes est très souvent conservé.
4) Doubler ou dédoubler les lettres ne changent pas la clé.
5) La plupart des transpositions ne changent pas la clé.

Les inconvénients les plus importants de cette clé découlent des poids attribués, dans la détermination de la clé, à la première lettre du mot et aux premières consonnes . En particulier, si l'erreur porte sur la première lettre, la distance des deux clés rend généralement impossible la correction ; par exemple le mot astre (clé ASTRE) ne peut être retrouvé à partir du mot faux satre (clé STRAE)

Exemple de performance :
Le programme SPEEDCOP (travaux de Pollock) corrige 85 à 95 % des erreurs pour lequel il a été prévu ce qui représente de 65 à 80 % des erreurs trouvées.

2.2.1.2.6 Les alphagrammes (Debili 86)

Cette méthode repose sur le fait que dans certaines langues (français, anglais, arabe), les anagrammes sont rares. La clé de similarité appelée *alphagramme* est constituée en concaténant l'ensemble des lettres constituant le mot dans l'ordre alphabétique.

Exemple d'alphagrammes :

papier	aeippr
porte	eoprt
arc	acr
car	acr
permettre	eeemprrtt

Selon (Debili 86) l'utilisation des alphagrammes comme clés permet de corriger
la suppression d'un ou de deux caractères,
la transposition de deux caractères adjacents,
la substitution d'un caractère par un autre ,
l'insertion d'un ou deux caractères,
et pour un nombre de cas plus ou moins réduits :
la conjugaison transposition + insertion
la conjugaison transposition + omission.

2.2.1.2.7 Les méthodes phonétiques

2.2.1.2.7.1 Principe

Cette méthode (Strube de Lima 90) trouve son origine dans le fait que, par suite d'une compétence insuffisante, l'utilisateur écrit "comme ça se prononce". Ce type d'erreurs peut être rattaché à certaines classes d'utilisateurs, notamment ceux en train d'apprendre la langue (scolaires, étrangers etc).

Etant donné une forme incorrecte E on détermine la représentation phonétique C de E appelé **clé phonétique**. A partir de C, on recherche toutes les formes du lexique ayant même clé phonétique. On applique éventuellement des critères restrictifs sur les solutions. Les solutions restantes sont proposées à l'utilisateur.

Nous allons examiner successivement les deux premières étapes.

2.2.1.2.7.2 Détermination de la clé

Cette étape s'appuie sur le transducteur d'état finis de PILAF, avec utilisation d'un dictionnaire de pseudo-graphèmes (p-graphèmes), qui ont été définis de manière à s'adapter à une bonne transcription, et, qui ne correspondent pas exactement aux graphèmes élémentaires du français.

L'analyse d'un mot consiste à déterminer un ou plusieurs découpages de ce mot en p-graphèmes en utilisant des règles et des modèles de nature phonétique.

Nous remarquerons qu'à un mot donné peuvent correspondre plusieurs clés et qu'inversement à une clé donnée peuvent correspondre plusieurs mots.

Exemples :

Graphie	Clé(s)				
vinku	v.in.k.u				
okurransse	o.k.u.r.an.s				
peti			p.é.t.i	p.e.t.i	
echo	e.ch.o	e.k.o.	é.ch.o.	é.k.o.	

2.2.1.2.7.3 Recherche des formes phonétiquement équivalentes

Deux méthodes ont été envisagées. La première repose sur l'utilisation d'un analyseur phonétique (analyseur morphologique de PILAF) et suppose la construction d'un dictionnaire de bases phonétiques équivalent au dictionnaire de bases morphologiques. Cette construction devant être manuelle, le coût en aurait été très important.

La seconde méthode n'utilise pas le transducteur de PILAF mais un ensemble de règles de correspondance entre les p-phonèmes et les p-graphèmes. A chaque p-phonème, on fait correspondre la liste des p-graphèmes qui peuvent le représenter. Un triplet de booléens associé à chaque p-graphème précise si ce p-graphème peut être une représentation du p-phonème en début, milieu ou fin de mot.

Exemples de correspondances p-phonème p-graphèmes :

Règles associées au p-phonème "*m*"			
p-graphème	début	milieu	fin
m	oui	oui	oui
me	non	non	oui
m.m	non	oui	non
m.me	non	non	oui

Pour éviter une combinatoire trop élevée, Strube de Lima propose deux améliorations :

1) L'utilisation de liste de cas particuliers permettant de restreindre les règles et de donner directement les graphies possibles (par exemple à la clé *i.l.o.* va correspondre les graphies *îlot* et *îlots*).

2) Analyse du dictionnaire de formes pour produire une table annexe d'adjacence de bi-grammes de graphèmes. Cette table est utilisée pour réduire le nombre de combinaisons au moment de la génération des chaînes graphiques.

2.2.1.2.7.4 La génération des graphies

La génération des graphies correctes, à partir des clés phonétiques associées, comprend quatre étapes :

1) Pour chaque chaîne phonétique, on commence par vérifier si ce n'est pas un cas particulier. Auquel cas la table spéciale permet d'obtenir directement la graphie exacte et la génération est terminée.
Dans le cas contraire
2) Pour chaque p-phonème de chaque chaîne phonétique, on calcule la liste des p-graphèmes possibles en utilisant les règles de correspondance.

3) On assemble pour chaque chaîne phonétique toutes les combinaisons possibles des p-graphèmes en vérifiant pour chaque paire que l'assemblage est possible (utilisation de la table d'adjacence).
4) La liste des graphies obtenues est filtrée par consultation du dictionnaire de formes.

2.2.1.2.7.5 Conclusion

Les corrections obtenues par cette méthode sont de bonne qualité même si les temps de réponse sont élevés. Cela s'explique par le nombre élevé de combinaisons possibles. Par exemple, en partant de la forme incorrecte *alez* on obtient 30 graphies (*allai, allaient, ..., halai, halaient, hâlais, ...*).

2.2.1.2.8 Les méthodes de génération morphologique

Cette méthode (Cohard 88) se place au niveau de l'interprétation des éléments constitutifs d'un segment. Elle a pour objectif de corriger les erreurs liées à une mauvaise utilisation des désinences.(exemples : *chevals, journals, faisez, ...)*

L'idée consiste à déterminer avec l'analyseur morphologique, un maximum d'informations (genre, nombre, temps, ...) sur la forme erronée puis d'utiliser le générateur morphologique pour générer le mot correct.

Pratiquement, l'ordre des opérations est le suivant :

1) On extrait la plus longue racine correcte du mot faux (par exemple *all* pour *allerons*, *chev* pour *chevals.*
2) On fait l'analyse morphologique du "reste" du mot pour obtenir un certain nombre d'informations. Par exemple première personne du pluriel du futur de l'indicatif pour *"erons'* et masculin pluriel pour "*als"*.
3) On utilise alors le générateur morphologique sur la racine en prenant comme valeurs des variables morphologiques celles calculées à l'étape 2 pour le mot faux.

On corrige ainsi *allerons* en *irons* et *chevals* en *chevaux.*

2.2.2 LE NIVEAU SYNTAXIQUE

2.2.2.1 Taxinomie des erreurs syntaxiques

Le classement ci-dessous est extrait de (Pécate 92). Dans le chapitre 7 sur les structures nous reviendrons sur certaines des règles d'accord en les détaillant.

2.2.2.1.1 Erreurs d'accord

Bien que comportant de nombreuses exceptions, les règles d'accord sont parfaitement définies dans la grammaire française. Elles sont réparties en trois catégories :

- Accord du verbe en nombre, en personne et en genre (pour les temps composés avec son sujet.
Exemple : *Les chevaux galope / les chevaux galopent.*

- Quelle que soit sa fonction (épithète, attribut, opposé), accord de l'adjectif qualificatif, en genre et en nombre, avec le nom ou le prénom auquel il se rapporte.

Exemple : *C'est un belle homme / c'est un bel homme*

- **Accord du participe passé** employé seul ou avec un auxiliaire (être, avoir).
Exemple : *Elles seront puni / Elles seront punies.*

Certains accords demandent non seulement une analyse syntaxique mais également une prise en compte d'informations sémantiques ; par exemple (Bescherelle)

la pianiste que j'ai entendue jouer; (Accord entre *la pianiste* et l'agent *entendre*).
la sonate que j'ai entendu jouer, (Pas d'accord car *la sonate* est un complément de *jouer*).

2.2.2.1.2 Erreurs d'emplois

2.2.2.1.2.1 Emploi de la forme d'un mot

La structure de la phrase est correcte mais le choix de la forme du mot s'est fait en désaccord avec les règles de la grammaire. On peut diviser ces règles en règles générales multi-mots et en règles spécifiques pour un mot.

Exemples :

- **Emploi de certaines formes du déterminant et de certains adjectifs devant un mot commençant par une voyelle ou un "h" muet** (*sa amie / son amie; ce homme / cet homme; beau homme / bel homme*).

- **Emploi de certaines formes de l'impératif : les verbes en "er" n'ont pas de "s" à la deuxième personne de l'impératif sauf devant un "y" et "en" non suivis d'un complément** (*Vas dans le champ / Va dans le champ; Va y / Vas-y; Va y chercher du repos / Vas y chercher du repos*).

- **Emploi des modes et des temps : la grammaire donne une règle de concordance des temps et de certaines formes** (Le "si" n'est jamais suivi du futur ou du conditionnel, le présent en tient lieu : *Si je recevrai des nouvelles / si je reçois des nouvelles*).

- **Emploi du bon auxiliaire (***Je suis été malade / J'ai été malade***).**

2.2.2.1.2.2 Emploi des conventions typographiques

La ponctuation

Citons quelques règles :

- utilisation de la virgule pour délimiter les groupes fonctionnels (apposition, détachement, ...)
Exemple : *Belles, elles attirent tous les regards.*
- équilibrage des parenthèses, des accolades, des guillemets.
- utilisation des espaces : seuls les signes composés de plus d'un élément sont précédés et suivis d'un espace

Exemples d'erreurs :

- ponctuation manquante : *Le garçon mange une pomme Elle est rouge*
- ponctuation mal placée : *Il est interdit de jouer au ballon, avec les pieds sur la plage.*
- mauvaise ponctuation à la fin d'une phrase ayant un sens interrogatif, exclamatif, affirmatif : *Quel beau temps nous avons eu hier ?*

Les majuscules
Il existe un ensemble de règles, les unes faciles à mettre en œuvre (par exemple majuscule en début de phrase et de noms propres) d'autres plus difficiles car reliées à de la pragmatique (Par exemple, première lettre en majuscule pour les noms de mois s'il s'agit d'une fête (*13 août* mais le *15 Août*).

Divers
De nombreuses règles existent, issues d'une évolution lente de la langue.
Exemple : t de liaison rajouté aux verbes terminés par une voyelle à la troisième personne du singulier dans tous les cas d'inversion (*A-t-il fait son travail ?*).

2.2.2.1.3 Erreurs de construction

Insertion d'un mot :
Cette faute se traduit en général par la répétition d'un mot ; exemple : *le bébé la la tète.* En français, les mots que l'on peut doubler sont rares (par exemple : les pronoms *nous* et *vous. nous nous rencontrons ...)* mais des exemples célèbres nous montrent l'impossibilité d'en définir la liste (*les poules du couvent couvent*).

Omission d'un mot :
Exemple *Elle le lui / Elle le lui dit.*

Transposition d'un mot :
Exemple : *Elle lui le dit.*

Confusion de deux mots entre eux :
Cette erreur se rencontre à l'occasion de mots phonétiquement équivalents mais dont l'usage syntaxique est différent.
Exemple 1 :

> Les couples :*a / à, et / est, on / ont, son / sont, ou / où, dont / don, ce / se, ces / ses.*

Exemple 2 :

> Egalité phonique du couple participe passé et infinitif des verbes du premier groupe.
> *Le professeur a démontrer le théorème,*
> *Ils son les meilleurs,*
> *Il est aller mangé.*

Fautes d'orthographes et/ou d'édition :
Un mot peut produire un autre mot, correct d'un point de vue lexical, mais pouvant posséder, comme pour la confusion vue ci-dessus, des fonctions syntaxiques différentes.

Exemples :

> *le mois de mais est bien gai ;*
> *je suis content que tu fosses la vaisselle.*

Si la fonction des deux mots est identique, seule la sémantique ou la pragmatique peut les différencier.

2.3 CHOIX RETENUS POUR LE SYSTÈME CÉLINE

2.3.1 INTRODUCTION

En absence de convention reconnue, nous allons tout d'abord préciser au paragraphe suivant le vocabulaire utilisé (principalement : texte, phrase, mot, graphie, forme textuelle ...). Abordant ensuite le système CELINE, nous présenterons nos choix sur quatre points clés :

- Face à une forme textuelle inconnue, nous présenterons les traitements choisis générant des formes textuelles candidates à un remplacement.
- Par définition du système recherché, pour aboutir au niveau de coopération des différents agents, le système se doit de ne pas isoler les traitements des erreurs au niveau lexical, au niveau syntaxique ou au niveau sémantique. Le système doit gérer une structure de données permettant la mise en attente des résultats de manière à pouvoir différer la prise de décision sans s'arrêter à la première solution. Le système doit prendre sa décision seulement lorsque l'ensemble des éléments lui est disponible. L'explosion combinatoire déjà importante demande de ne pas refaire les mêmes analyses. Concrètement le système va travailler phrase après phrase et va conserver l'ensemble des résultats relatifs à une phrase jusqu'à la prise d'une décision définitive pour cette phrase.
- Réciproquement cette structure constitue en même temps une source de données consultable par les agents du système si leurs accointances leur autorisent cette vision. Dans le chapitre 9 sur les systèmes multi-agents, nous reviendrons sur cet aspect du fonctionnement et nous élèverons cette structure de données au rang de tableau noir. La coopération spontanée ou déclenchée des autres agents du système va permettre peu à peu de préciser les solutions acceptables et celles à éliminer.
- L'importance de cette structure de données va demander un gros effort de gestion de celle-ci. Nous proposons de gérer cette structure en utilisant un SGBD. Poursuivant dans ce sens, un agent central « traitement de texte détecteur et correcteur d'erreurs » pourrait alors être conçu sous la forme d'un SGBD travaillant en coopération avec d'autres agents experts. La phrase corrigée apparaîtrait alors comme une vue extraite des tables.

2.3.2 TERMINOLOGIE GÉNÉRALE

2.3.2.1 Définitions relatives au corpus à traiter

- *Rédacteur* : l'utilisateur du système saisissant le texte.
- *Délimiteurs de fin de phrase* : caractères particuliers tel que . ! ? …
- *Phrase* : portion de corpus comprise entre deux délimiteurs de fin de phrase.
- *Mot* : unité du discours.
- *Forme* : graphie d'un mot délimité par des blancs.
- *Forme textuelle* : forme rencontrée dans un corpus.
- *Forme lexicale (base)* : partie d'une forme textuelle qui reste inchangée dans n'importe quel contexte.

- *Paramètres linguistiques* : l'étude d'une langue naturelle amène à définir :
  - Des *paramètres morphologiques* : ils décrivent le comportement morphologique des formes textuelles.
  - Des *paramètres syntaxiques*
  - Des *paramètres sémantiques*
- *Catégorie morphologique* : à chaque mot correspond une et une seule catégorie morphologique qui désigne le fonctionnement morphologique du mot dans la langue quel que soit le contexte. A une même graphie peuvent correspondre différentes catégories.
- *Variables morphologiques* : paramètres morphologiques qui décrivent le statut d'un mot dans un contexte donné (genre, nombre...).
- *Règles morphologiques* : règles permettant, par analyse de la forme textuelle, d'affecter un certain nombre de paramètres morphologiques.
- *Lexique* : ensemble fini B de bases lexicales auxquelles on associe :
  - les paramètres morphologiques correspondants,
  - les règles à activer,

  ce qui traduit le comportement de la base considérée.

2.3.2.2 Définitions provisoires relatives aux systèmes multi-agents

Nous découvrirons par la suite la terminologie des systèmes multi-agents. Nous utiliserons provisoirement les mots dont les définitions suivent :

- *Agent* au sens provisoire de module de traitement.
- *Agent éligible* : agent connu du système mais non encore sollicité pour une tâche.
- *Agent élu* : agent lancé par le système.
- *Agent élu actif* : agent en train de travailler.
- *Agent élu inactif* : agent sans travail.

2.3.3 TRAITEMENT D'UNE FORME TEXTUELLE INCONNUE

Ce traitement va comprendre trois parties :

1. La proposition de nouvelles formes textuelles remplaçantes, par diverses méthodes en restant dans la lignée de DECOR et CORSYN :
   - traitement alphabétique,
   - concaténation de formes textuelles consécutives inconnues,
   - génération morphologique,
   - génération phonétique,
   - modélisation de Markov des chaînes lexicales d'un réducteur particulier,
   - alphagramme de la table lexicale d'un rédacteur particulier,
   - interfaçage avec un des correcteurs classiques des traitements de texte (par exemple WORD). Le correcteur fournit dans ce cas les formes textuelles remplaçantes sans les éléments morphologiques (catégories et variables), syntaxiques ou sémantiques.

2. Elimination/validation d'un certain nombre de formes concurrentes à partir de la graphie proposée (le procédé de validation doit être différent de celui ayant amené la proposition).

- reconnaissance par clé squelette,
- reconnaissance par analyse lexicale,
- distance de corrections,
- modélisation de Markov,
- alphagramme...

3. Elimination d'un certain nombre de formes concurrentes par des contraintes syntaxiques ou sémantiques au niveau de la structure dans laquelle est située cette forme textuelle. La structure est, elle même, contrainte par l'ensemble de la phrase.

2.3.3.1 Traitement alphabétique : suppression, ajout, permutation de lettres

Tous les traitements proposés sont facultatifs. A l'initialisation du système, le paramétrage permet de faire le choix des méthodes retenues ainsi que de l'ordonnancement de leurs applications. Ensuite, par le biais de la statistique, le système va évoluer librement et l'ordonnancement tiendra compte de l'efficacité de chaque méthode.
Pour ces traitements alphabétiques, les méthodes mises en œuvre pour l'instant supposent :

- une seule faute sur la racine de la forme textuelle (plus par exemple une faute d'accord),
- pas d'arrêt à la première solution trouvée et validée sur l'ensemble des niveaux. Le système va essayer de rechercher plusieurs solutions.

Ordonnancement (par défaut) des traitements alphabétiques	
Faute supposée	Traitement
Lettre triple	On supprime une des trois lettres et en cas d'un nouvel échec de l'analyse on en supprime deux. (Exemple : tétttard pour tétard)
Lettre double	On en supprime une (Exemple : propposition pour proposition)
Permutation de deux lettres	On permute les lettres adjacentes deux à deux (Exemple : volie pour voile; on va essayer successivement : ovlie, vloie, voile)
Lettre manquante	On essaye successivement les 26 lettres à toutes les positions possibles. (Exemple : accient pour accident, on va essayer Xaccient avec X égal successivement à a,b,c,d, ...x,y,z et é à û î puis aXccient, acXcient, accXient, acciXent)

Dans les essais de lettres, on peut prendre l'ordre alphabétique normal mais on peut aussi utiliser des chaînes de Markov des caractères permettant d'optimiser les essais. Si on ne s'arrête pas à la première solution trouvée, de toute façon toutes les lettres vont être essayées et le temps de recherche va être le même. Par contre, dès qu'une solution est trouvée, elle va être prise en compte et exploitée par le système. Globalement on gagne donc du temps.

Les formes textuelles proposées par les traitements alphabétiques vont ensuite être rejetées ou validées par un lexique qui y ajoutera dans ce cas catégories et variables morphologiques.

2.3.3.2 Formes textuelles inconnues adjacentes

Lorsque deux formes textuelles successives d'une même phrase ne sont pas reconnues, on peut penser quelles proviennent de la coupure néfaste d'une forme unique. On peut alors essayer diverses solutions :

- concaténation des deux formes textuelles
- concaténation de : première forme textuelle, une lettre X et la deuxième forme textuelle avec X égal successivement à a,b,c,d, ...x,y,z. Là encore le choix du caractère à ajouter peut être optimisé par les modèles de Markov ou par la disposition du clavier.

> Exemple : *cani he*.
> 1. Par concaténation on obtient *canihe* mot inconnu.
> 2. Par concaténation et insertion : si on examine les lettres voisines de la barre d'espacement on trouve *x, c, v, b, n*. on génère *canixhe, caniche, canivhe, canibhe, caninhe.* Une seule solution *caniche* va être ensuite validée par la reconnaissance lexicale.

2.3.3.3 La génération morphologique

L'analyse morphologique initiale du mot inconnu, bien qu'ayant échouée, peut avoir fourni des indices :

- racine(s) possible(s),
- signes sur les variables morphologiques (genre, nombre, etc.).

Différents agents (PSCM par la modélisation de Markov; PAS par l'analyse syntaxique) peuvent proposer de leur côté une ou plusieurs catégories morphologiques.

Le module de vérification des accords peut fournir, en s'appuyant sur les structures reconnues, des renseignements sur les variables morphologiques ou même sur les catégories morphologiques.

Au total, si on dispose de l'ensemble « racine+catégorie morphologique + variables morphologiques » on peut alors demander au générateur morphologique de proposer une solution.

2.3.4 ELIMINATION DE SOLUTIONS CONCURRENTES

L'étape précédente ayant fourni un ensemble de formes textuelles candidates, il faut maintenant les valider ou les refuser. Un certain nombre d'agents sont mis en concurrence pour désambiguïser le choix des formes textuelles candidates. Nous verrons que les avis proposés par les agents sont affectés de divers coefficients permettant un choix final quantitatif si possible unique.

Parmi les procédés utilisables citons :

- La reconnaissance lexicale (nous venons de voir un exemple au § 2.3.3.3)
- La reconnaissance par clé squelette
- La reconnaissance phonétique
- Les modèles de Markov qui vont pouvoir proposer pour le mot inconnu une ou plusieurs catégories morphologiques. Les mots possédant cette catégorie vont être confirmés.
- La vérification des accords permettant de valider ou d'éliminer une forme candidate à partir des variables morphologiques qui lui sont rattachées

- Des techniques liées à la connaissance du rédacteur : distance de correction, erreur de frappe, etc.

Evidemment le ou les procédés de validation ne doivent pas être le ou les mêmes que celui ou ceux ayant proposé la forme candidate. Pour chaque proposition, il y a donc lieu de mémoriser les agents proposant cette forme.

Le paragraphe suivant va nous permettre d'illustrer cette démarche. La structure de données qui va se dessiner petit à petit, va utiliser nos suggestions et remarques du paragraphe sur la méthode des cartes.

2.3.5 APPROCHE DE LA STRUCTURE DE DONNÉES NÉCESSAIRES

2.3.5.1 Initialisation de la structure de données

1) Le texte est découpé en phrases Le traitement s'effectue phrase par phrase. Pour la suite nous suivrons un exemple support : *je manje une lomme fraîche.*

2) Les phrases sont transmises au secteur lexical qui active un certain nombre d'agents lexicaux. Les agents lexicaux valident les formes qu'ils reconnaissent et renvoient catégories et variables morphologiques correspondantes. Les formes résiduelles non reconnues des lexiques peuvent donc être considérées dans un premier temps comme des erreurs que le système va chercher à résoudre en rentrant dans un cycle propositions / validations / réfutations. Si le système n'y parvient pas, alors en dernier recours, il fera appel à l'agent humain qui décidera s'il s'agit d'un mot inconnu qu'il validera ou d'une erreur incorrigible par le système et qu'il corrigera de lui-même.

3) Le secteur lexical va chercher à formuler des propositions de remplacement en activant les agents possédant cette faculté.

Dans le cas de notre exemple, *manje* et *lomme* n'étant pas reconnus par les lexiques, un interfaçage avec Word 6 sous Mac nous permet d'obtenir 7 propositions pour *manje* et 8 propositions pour *lomme* :

Forme textuelle 1	Forme textuelle 2	Forme textuelle 3	Forme textuelle 4	Forme textuelle 5
je	*mange* *manges* *manne* *manie* *mande* *manse* *mante*	*une*	*somme* *sommes* *comme* *gomme* *homme* *nomme* *pomme* *tomme*	*fraîche*

En partant de la graphie de la forme textuelle remplaçante, les agents lexicaux fournissent alors les catégories morphologiques et les variables associées et complètent ainsi la structure de données.

5) Dans le cas d'un mot inconnu (et non d'une erreur) l'agent humain validera la forme textuelle. Dans ce cas, les agents (modèles de Markov, analyses syntaxiques, vérifications des accords …) chercheront à compléter, sans lien avec la graphie, catégories et variables morphologiques de la forme inconnue.

6) Dans le cas de notre exemple, nous voyons que les propositions émises permettent d'envisager un ensemble de 7 x 8 = 56 combinaisons différentes.
Ces combinaisons vont être mémorisées dans la structure de données sous la forme de lignes multiples. Pour chaque forme textuelle candidate, on va engendrer autant de lignes que de combinaisons possibles catégorie morphologique + variables morphologiques.

7) Les agents d'analyses syntaxiques vont commencer à travailler et vont compléter le tableau avec les groupements syntaxiques qu'ils reconnaissent et que nous appellerons *structures.*

> Exemple :
> Un analyseur syntaxique va alors reconnaître la séquence *pronom verbe* comme un syntagme verbal SV puis la séquence *déterminant substantif* comme un groupe nominal GN. Ce travail sera mémorisé sous la forme de listes parenthésées du type arbre en LISP. Par exemple la notation SV(1.0, 2.1) fait référence à la forme textuelle unique n° 1 de la phrase et à la première forme textuelle candidate pour la position n° 2 de la phrase.

Dans le tableau ci-après nous présentons en extension les structures. Dans la réalité les structures sont repérées par un identifiant et placées dans un deuxième tableau. D'un point de vue syntaxique, le travail sur ce tableau est tout à fait comparable à celui qui se fait dans la méthode des Cartes (Menézo 92) et nous retrouvons l'équivalent de la table des chaînes bien formées. Du point de vue de CELINE, certaines structures vont être réfutées par des contraintes imposées par les autres agents (Markov, Accord, etc.)

Structure de données partielles					
Position dans la phrase	N° d'ambiguïté	Forme textuelle	Catégorie morphologique	Variables Morphologiques	Structures (terminales)
1	1.0	je	pronom personnel	Première personne	SV(1.0, 2.1) SV(1.0, 2.2) SV(1.0,2.4) SV(1.0, 2.5)
2	2.1	mange	verbe	Première personne Présent Indicatif	SV(1.0, 2.1)
2	2.2	manges	verbe	Deuxième personne Présent Indicatif	SV(1.0, 2.2)
2	2.3	manne	substantif	Féminin Singulier	
2	2.4	manie	verbe	Première personne Présent Indicatif	SV(1.0, 2.4)
2	2.5	mande	verbe	Première personne Présent Indicatif	SV(1.0, 2.5)
2	2.6	manse	substantif	Féminin Singulier	R
2	2.7	mante	substantif	Féminin Singulier	
3	3.0	une	article indéfini	Féminin Singulier	GN(3.0, 4.1, 5.0) GN(3.0, 4.2, 5.0) GN(3.0, 4.5, 5.0)

					GN(3.0, 4.6, 5.0) GN(3.0, 4.7, 5.0) GN(3.0, 4.8, 5.0) GN(3.0, 4.9, 5.0)
4	4.1	somme	substantif	Féminin Singulier	GN(3.0, 4.1, 5.0)
4	4.2	sommes	substantif	Féminin Pluriel	GN(3.0, 4.2, 5.0)
4	4.3	sommes	verbe	Deuxième Personne Présent Indicatif	
4	4.4	comme	conjonction		
4	4.5	gomme	substantif	Féminin Singulier	GN(3.0, 4.5, 5.0)
4	4.6	homme	substantif	Masculin Singulier	GN(3.0, 4.6, 5.0)
4	4.7	nonne	substantif	Féminin Singulier	GN(3.0, 4.7, 5.0)
4	4.8	pomme	substantif	Féminin Singulier	GN(3.0, 4.8, 5.0)
4	4.9	tomme	substantif	Féminin Singulier	GN(3.0, 4.9, 5.0)
5	5.0	fraîche	adjectif	Féminin Singulier	GN(3.0, 4.1, 5.0) GN(3.0, 4.2, 5.0) GN(3.0, 4.5, 5.0) GN(3.0, 4.6, 5.0) GN(3.0, 4.7, 5.0) GN(3.0, 4.8, 5.0) GN(3.0, 4.9, 5.0

Remarque : pour simplifier le tableau ci-contre :

1. Seules les structures terminales ont été portées et non les structures non terminales du type (P(SV(1.0,2.1),GN(3.0,4.1,5.0))).
2. Certaines indications que nous verrons ultérieurement n'y figurent pas (marques de validités, qualités des structures, etc.).

8) A partir du moment où une structure apparaît dans le tableau, les agents de vérification des accords vont pouvoir commencer à travailler et ainsi valider, refuser ou compléter les variables morphologiques.

2.3.5.2 Marques de validité

2.3.5.2.1 Principe des marques de validité

Tous les agents actifs vont pouvoir donner leur avis sur les formes textuelles, les catégories morphologiques et les variables associées. Une marque appelée *marque de validité* va permettre à chaque agent appelé à donner son avis de :

- Détecter les avis à donner pour lesquels il est sollicité.
- Donner effectivement son avis codifié selon son domaine de compétence sur tel ou tel aspect du tableau.

Forme textuelle inconnue

⇓

1	Forme textuelle substituante 1	Marque de validité de la forme textuelle	Catégorie morphologique	Marque de validité de la catégorie	Variable morphologique	Marque de validité des variables

		Px, Rx, Vx, Ix	possible	Px, Rx, Cx, Ix, Vx	associée	Px, Rx, Cx, Ix, Vx
2	Forme textuelle substituante 2		Catégorie Morphologique		Variable morphologique associée	
3	Forme textuelle substituante 3		Catégorie morphologique		Variables morphologique associée	
...	...	...	...			...

La marque de validité indique la « position » de la forme textuelle, de la catégorie ou de la variable candidate :

- P : la forme textuelle, la catégorie ou la variable est proposée.
- C : la forme textuelle, la catégorie ou la variable à été confirmée
- R : la forme textuelle, la catégorie ou la variable est définitivement refusée.
- I : Incapacité de l'agent à traiter le problème.
- V : la forme textuelle, la catégorie ou la variable est complètement validée et définitivement retenue (dans un vecteur de mot remplaçant, un seul mot peut être validé).

Les procédés sont repérés par des identifiants à deux niveaux :

- une lettre repère la classe de l'agent
- un numéro donne l'identification de l'agent

Exemple
a0 représente le correcteur de WORD, b0 représente le lexique PILAF, b1 le dictionnaire personnel, b2 le lexique de le chimie organique. Pb0 représente une proposition par le correcteur de word, Vb1 représente une validation par PILAF

Procédé x	
a	Correcteurs indépendants
b	Lexiques
c	Modélisation de Markov.
d	Clé phonétique.
e	Clé squelette.
f	Analyse syntaxique.
g	Génération morphologique.
i	Analyse sémantique.
j	Vérification des accords.

2.3.5.2.2 Exemple

Examinons le travail des agents sur la portion de phrase candidate *je manges.*

Proposition pour la forme textuelle 2 *manje*					
Forme textuelles candidates	Validité FT	Catégorie morphologique	Validités CM	Variable morphologique	Validités VM
manges	Rj	verbe	Cc Cf Cg	2 ième personne	Rj

La catégorie *verbe* est exacte et les agents appelés à donner leur avis la reconnaissent. Après que l'agent d'analyse syntaxique ait proposé la structure SV (1.0, 2.0), la variable morphologique *deuxième personne* va être refusée par l'agent de vérification des accords. Un refus de la variable morphologique ou de la catégorie morphologique

entraîne une marque de refus R pour la forme textuelle proposée. Dès que cette forme textuelle est refusée, toutes les structures où elle intervient sont rayées du tableau.

Évidemment pour la portion de phrase *je mange* nous obtiendrons cette fois la validation de la variable morphologique. Comme il n'y a pas par ailleurs de refus de la catégorie, la forme textuelle peut être validée.

Proposition pour la forme textuelle 2 *manje*					
Forme textuelles candidates	Validité FT	Catégorie morphologique	Validités CM	Variables morphologiques	Validités VM
mange	Vj	verbe	Cc Cf Cg	1 ere personne	Vj

Nous avions 56 phrases candidates. Le travail des agents amenant chacun ses éléments va permettre petit à petit de refuser certaines d'entre elles. Pour la phrase *je manges une homme,* l'agent de vérification des accords aura refusé *manges* pour sa deuxième personne et *homme* pour son genre masculin.

Formes textuelles initiales	je	manje	une	lomme
Formes textuelles proposées	je	manges	une	homme
Etat de la phrase	R			
Etat des formes textuelles	V	R	V	R
Catégories morphologiques	Pronom Personnel	verbe	pronom personnel	substantif
Validations de la C.M.	V	V	V	V
Variables morphologiques associées	Première personne	deuxième personne du présent de l'indicatif	féminin singulier	masculin singulier
Validation de la VA	V	R	V	R

Un module de sémantique ou un module de « styles et habitudes de parole » va refuser une *somme fraîche*, une *gomme fraîche*, une *nonne fraîche*.

Nous pouvons remarquer que

- Par la gestion de cette structure, nous obtenons notre objectif défini dans le chapitre d'introduction : la fusion des étapes de proposition, de reconnaissance, d'analyse syntaxique, de vérification des accords en travaillant sur de petits groupes de mots qui vont être successivement proposés, réfutés ou validés par les différents agents.
- Si nous disposons d'un agent d'analyse syntaxique travaillant directement sur cette structure, cet agent va indirectement bénéficier des contraintes imposées par les autres agents et va effectuer une analyse syntaxique contrainte. Il s'agit donc bien d'un travail en coopération guidé par les données.

2.3.5.3 Auto-optimisation du traitement

Nous voyons qu'en terme de rapidité d'élimination des fausses solutions sur la phrase, il serait dans ce cas intéressant de commencer par la prise en compte de la première personne du *je* et du féminin du *une*. Malheureusement dans ce cas, les indices sont trop minces pour détecter ce choix.

Nous déterminerons quelques règles stratégiques permettant de telle optimisation :

1. En utilisant s'ils existent *des îlots de confiance* sur la phrase. Ces îlots de confiance sont des zones sans erreurs que les agents vont identifier, structurer, et pour lesquels la structure va être en principe remplie en premier. En effet, les formes textuelles inconnues vont demander une recherche dans tous les lexiques. Après l'échec de leur reconnaissance, il faudra encore que le système fournisse les formes textuelles remplaçantes. Entre temps, les analyseurs auront eu le temps de travailler sur les formes textuelles adjacentes reconnues.
2. En utilisant des règles par rédacteur en fonction de ses compétences.

2.3.5.4 Règles de mise à jour de la structure de données

Récapitulons quelques règles de gestion de cette structure :

1. Quand un agent amène une modification sur un des attributs, on crée une nouvelle ligne avec la prise en compte de cette modification.
2. Quand un agent rempli une case blanche, toutes les marques de validité sont effacées pour que la phrase soit de nouveau complètement traitée ou retraitée par tous les agents.
3. Si une ligne est invalidée (forme textuelle rejetée), cela entraîne la suppression dans tout le tableau des structures comprenant la forme textuelle correspondante.
4. Tant que la marque de validité est P, tous les agents élus du système peuvent être amenés à modifier cette ligne.
5. Dès que la marque de validité est V, tous les agents élus du système considèrent cette information comme sûre et ne devant plus être remise en question.
6. Un refus net R émis par un des agents sur la variable morphologique ou sur la catégorie fait passer la marque de validité de la forme textuelle à R et fait refuser la ligne complète.
7. Les lignes refusées ne sont pas effacées afin que le système ne recommence pas les explorations déjà faites.

2.3.5.5 Problèmes de gestion de cette structure et de cohérence

Si cette structure de données est accessible à de nombreux agents, il faut alors gérer les problèmes de cohérence dans l'initialisation, la lecture, la mise à jour et la suppression de chaque ligne de la structure.
On peut envisager en première solution d'implanter les systèmes classiques lourds à gérer : sémaphores, moniteurs, …
Nous proposons une autre voie de recherche : utiliser comme structure de données des tables d'un agent SGBD dont les utilisateurs seront les différents agents du système : lexiques, analyseurs, générateurs…

2.3.6 INDUCTION D'UNE NOUVELLE DÉFINITION D'UN CORRECTEUR

La phrase (ou les phrases) est (sont) alors une (des) *vue(s)* construite(s) sur les tables. Le traitement de texte correcteur est alors le SGBD lui même (figure 2.11).

Chaque enregistrement aura quatre champs de clés :

1. le numéro représentatif de la place de la forme textuelle dans la phrase,
2. la forme textuelle proposée
3. la catégorie morphologique

4. Le jeu unique de variables morphologiques.

> Exemple *Le chien marron …*
> Si on admet pour *marron* soit l'adjectif (invariable par rapport genre et au nombre) soit le substantif, on créera les enregistrements :

Position	Forme textuelle	Catégories	Jeu de variables
3	marron	Adjectif	Féminin/Singulier
3	marron	Adjectif	Masculin/Singulier
3	marron	Adjectif	Féminin/Pluriel
3	marron	Adjectif	Masculin/Pluriel
3	marron	Substantif	Masculin/Singulier

Le SGBD assure deux fonctions clés :

- La non-redondance de ligne contenant un même quadruplet des quatre clés.
- La cohérence des informations.

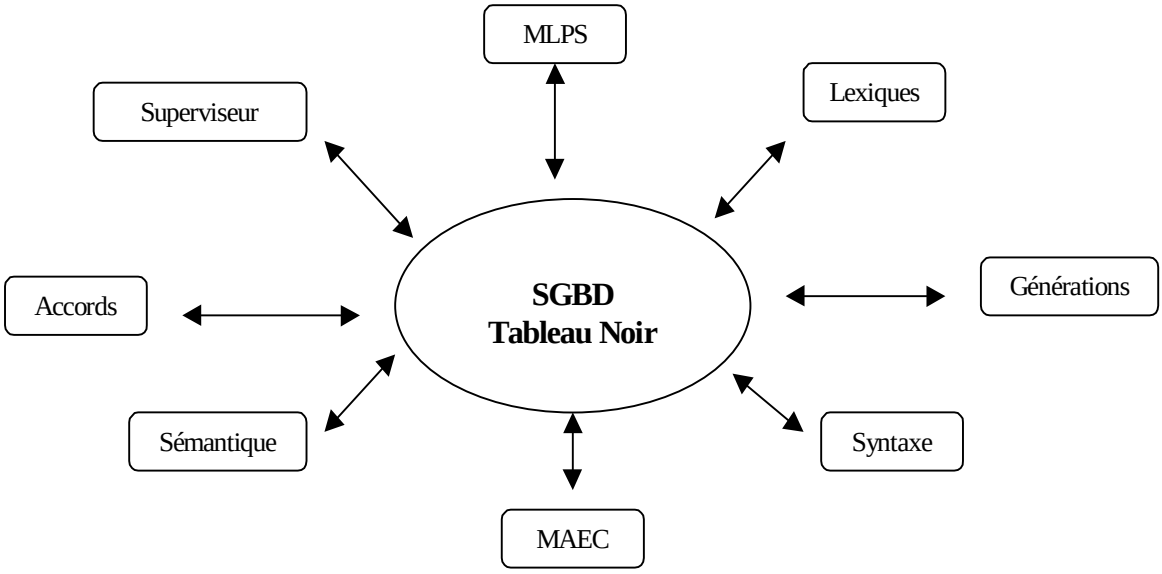


Figure 2.11

2.3.7 COÛT DES TRAITEMENTS

Certains des traitements mis en œuvre peuvent être assez coûteux en temps de travail. Nous avons décidé de ne pas tenir compte du coût pour trois raisons essentielles :

1. Le coût algorithmique est polynomial. Ocello (Occello 91) a modélisé un système tableau noir par un réseau de Pietri pour définir un mécanisme de contrôle générique. S'intéressant ensuite aux coûts (y compris fréquence du contrôle, duplication moyenne des agents …), Ocello (Ocello 93) reprend les travaux de Carlier et Chrétienne (Carlier 88) et démontre que pour un tableau noir parallèle la complexité du contrôle est en $\theta(n^2)$ n étant le nombre d'agent. Nous montrons que notre structure et l'intervention de nos agents correspondent bien à un tableau noir parallèle.

2. Cet aspect est relativement secondaire dans le cas d'un traitement automatique qui s'effectue en tâche de fond sans interaction avec l'agent humain.

3. L'évolution des machines est telle, qu'à tâche identique, l'ordre de grandeur du temps nécessaire est divisible par un facteur de 2 à 5 tous les ans. Citons la revue *Sciences et Vie Micro* de novembre 97 (pp 79-82) :
    - *La rapidité à laquelle évolue les processeurs est telle que de ceux qui étaient en vente en 1995, plus aucun n'est fabriqué aujourd'hui ... Ne choisissez pas un modèle Pentium inférieur au pentium MMX 166 Mhz, il serait trop peu performant pour les applications de demain.*

Si nous regardons l'avenir proche nous pouvons être rassurés ::

- *IBM début février 98 vient de franchir le cap mythique du Giga Hertz sur une puce expérimentale* (SVM Mars 1998 pp 44) .
- *Digital devrait commercialiser une puce de 15 millions de transistors à 600 Mhz puis à 850 Mhz début 1999. Même à fréquence d'horloge égale, ces nouvelles puces devraient être deux fois plus rapides* (SVM Mars 1998 pp 44) .
- *Le pentium III Katmai arrive sur le marché, de quoi faire passer les premiers Pentium II pour des antiquités* (PC Professionnel n° 14 – Janvier 1999).

2.4 CONCLUSION

Outre l'étude bibliographique très riche d'enseignements, ce chapitre nous a fait faire un pas fondamental dans l'analyse du système recherché. Nous déterminons petit à petit le cahier des charges de notre futur système de correction des erreurs. En partant de nos besoins de conservation d'informations nous avons commencé de définir la structure de données et en partant de nos besoins d'intervention sur la structure de données nous avons commencé à mettre en place ce que nous appellerons bientôt le protocole de communication. Nous continuerons cette spécification de CELINE à partir du chapitre 4. Auparavant un chapitre va nous présenter quelques réalisations du domaine.

3. PROTOTYPES DE CORRECTION DES ERREURS

Finalité

L'équipe TRILAN s'est toujours intéressée à la correction des erreurs. Nous pouvons en particulier penser aux travaux de Brigitte Cohard (Cohard 88), Vera Strube de Lima (Strube de Lima 89), Damien Genthial (Genthial 91).

Nous avons choisi quatre réalisations du domaine intéressantes sur des plans différents et dont les dates couvrent les dix dernières années :

- DECOR (Juin 88) intéressant sur le plan de la correction lexicale par les procédés mis en œuvre,
- CORSYN (Mars 90) qui prend en compte l'aspect syntaxique,
- VORTEX et VORTEXPLUS (Janvier 92) qui introduit une adaptation au rédacteur,
- ECLAIR (Décembre 93) qui travaille sur le plan lexical et prend en compte la notion de multi-lexiques.

Résumé

Ce chapitre va donc commencer par une synthèse rapide sur DECOR et CORSYN, réalisations dont les principes ont été déjà développés au chapitre 2.

Une autre réalisation ECLAIR (Letellier 93) fera l'objet d'une étude approfondie. Ce système se présente comme un système multi-experts et multi-lexiques d'analyse et de correction lexicales. Eclair est particulièrement intéressant car son cahier de charge, assez complet sur le plan du traitement orthographique, tient compte d'une possibilité de coopération avec un analyseur syntaxico-sémantique déterministe ANDI (Grandchamp 86), lui même intégré à CARAMEL un système multi-experts spécialisé dans le traitement automatique de la langue (et que nous retrouverons au chapitre 8 sur les systèmes multi-agents).

3.1 DECOR

3.1.1 TECHNIQUES UTILISÉES

DECOR (Cohard 88) repose sur l'utilisation de trois méthodes (clé squelette, clé phonétique, génération morphologique) que nous avons déjà présentées au chapitre 2 sur la problématique des erreurs.

Les trois méthodes sont plus ou moins bien adaptées à certains types d'erreurs, et de ce fait, possèdent des caractéristiques partiellement complémentaires.

DECOR utilise deux des combinaisons des trois méthodes précédentes : clé squelette + phonétique ou bien clé squelette + génération morphologique.

3.1.2 CRITIQUES

L'architecture repose sur une simple composition séquentielle. (Courtin 89) propose une architecture dans lequel un programme pilote effectue le choix de la ou des

méthodologies à utiliser, selon l'utilisateur, en fonction de son appartenance à une classe donnée d'utilisateurs (scolaire, étranger ...).

L'utilisation des trois méthodes demande un encombrement relativement important des tables utilisées (environ 7 MO) (Genthial 91).

La méthode de génération semble spécialement intéressante si on suppose une interaction des niveaux lexico-morphologique et syntaxique car, alors, les renseignements tirés de l'étude de la désinence fausse peuvent être appuyés ou même remplacés par ceux déduits de l'analyse syntaxique ou de l'unification des accord, qui, en cas de problèmes, pourraient proposer des variables morphologiques pour le mot erroné.

3.1.3 CONCLUSION

DECOR présente un intérêt certain par ses méthodes de correction lexicale basées sur les clés ou la génération morphologique que nous continuerons à utiliser.

3.2 CORSYN

Le prototype CORSYN (Strube de Lima 92) est composé de trois modules :

- un analyseur morpho-syntaxique qui regroupe l'analyseur lexico-morphologique et le constructeur de structures de dépendances de PILAF.
- un vérificateur syntaxique que nous allons détailler,
- un générateur morphologique qui est celui de PILAF.

3.2.1.1 Principe de la vérification syntaxique

L'entrée du vérificateur se présente sous la forme d'un arbre de dépendances décoré par des variables morphologiques.
Par exemple la phrase *Le chien rapporte la balle* se traduit par la structure (figure 3.1):

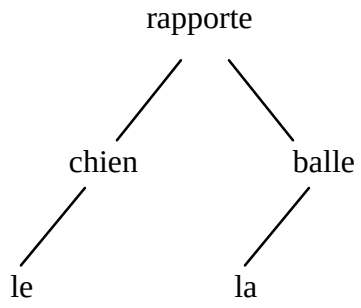


Figure 3.1 : arbre de dépendances

Le vérificateur parcourt l'arbre en ordre postfixé gauche, en examinant toutes les paires de nœuds (dépendant gauche, gouverneur) et (gouverneur, dépendant droit) (figure 3.2).

Pour l'exemple précédent, l'ordre d'examen est :

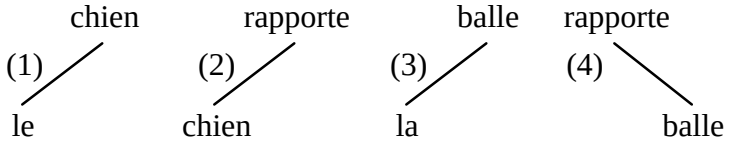


Figure 3.2 : paires de nœuds

Le vérificateur tente d'appliquer, à chaque paire rencontrée, un ensemble de règles de vérification.

Règles de vérification

Ces règles comprennent trois parties :

- Un entête indique la paire sur laquelle la règle peut s'appliquer ;
- Une partie conditionnelle SI qui indique les conditions que doivent respecter les variables morphologiques pour que la règle soit applicable.
- Une partie action ALORS qui indique comment ajuster les variables morphologiques.

Le vérificateur est écrit en PROLOG, les règles se traduisant par des prédicats.

Pour une paire (gauche*droit) donnée, le vérificateur n'applique que la première règle applicable. Une absence de règle applicable pour une paire donnée indique soit une incohérence entre la grammaire de dépendances de l'analyseur et la grammaire du vérificateur, soit que la structure est incorrecte. Dans les deux cas la vérification est abandonnée.

L'existence d'une règle pour toute paire suppose donc que certaines règles ne servent qu'à éviter l'arrêt de l'analyseur, l'obligation de ces règles est liée à l'utilisation du langage Prolog. Si n est le nombre de catégories morphologiques, le nombre de règles serait donc n^2. Pour diminuer le nombre de règles, les catégories utilisées pour l'analyse ont été regroupées en classes, chaque classe correspondant à un comportement homogène.

Métarègles de vérification

Ces règles ont pour objectif de regrouper dans une même règle les phénomènes d'accords communs. Par exemple :

(adjo * gnomo) --> (gnomo*adjo)
(gnomo*ppaso) --> (gnomo*adjo)

sont deux règles modélisant le fait que l'adjectif placé avant le nom ou le participe passé placé après le nom s'accordent de la même façon que l'adjectif placé après le nom.

3.2.1.2 Fonctionnement de CORSYN

Le fonctionnement du prototype CORSYN peut être résumé par le schéma algorithmique :

1) Entrée de la phrase supposée lexicalement correcte.
2) Construction d'une ou plusieurs structures de dépendances, correspondant à la phrase complète.

3) Cette structure (la première en cas de solutions multiples) est soumise au vérificateur syntaxique. Deux cas se présentent alors :
4) La structure ne comporte pas d'erreurs, le traitement s'arrête. On ignore les autres structures.
5) Le vérificateur détecte une erreur sur une paire (gauche*droit) :
6) Il transmet au générateur le mot à corriger avec les valeurs des variables à changer.
Le générateur produit alors une forme qui est proposée à l'utilisateur. Ce dernier a alors deux choix possibles :
7) Soit il accepte la correction : l'arbre est alors mis à jour et le traitement reprend à l'étape 3.
8) Soit il refuse la correction. Le traitement reprend alors à l'étape 3 avec la structure suivante.

3.2.1.3 Critiques de ce prototype

1) Le prototype suppose que l'analyseur fournit au moins un arbre complet. Cela élimine donc la possibilité de pouvoir traiter de nombreux cas d'erreurs qui se traduisent par une discontinuité de l'arbre syntaxique et la production d'une suite d'arbres. Ces erreurs pourront être d'origine orthographique et non syntaxique. Par exemple, la phrase *il **à** un chien* ne se traduit pas par un arbre complet.

2) Le choix du mot à corriger à l'étape 6, est fixé a priori pour deux catégories données. Par exemple, en cas de désaccord entre le nom et le déterminant, c'est le nom qui est soumis à correction.

Deux reproches peuvent être avancés en conséquence :
a) Ce type de règle manque complètement de souplesse et ne permet donc absolument pas, ni d'utiliser une stratégie de correction propre à un utilisateur donné, ni même d'utiliser des règles heuristiques comme celles déjà rencontrées.
b) De plus la faible granularité (deux mots) rend cette méthode assez "suspecte" dans le cas de structures imbriquées complexes.

3) Le parcours manque lui aussi de souplesse.

4) Il semble difficile, dans le cadre de cet algorithme, d'introduire des éléments sémantiques liant deux nœuds non adjacents.

3.2.1.4 Conclusion

CORSYN avait l'intérêt de proposer des corrections grâce au générateur morphologique (par exemple *les œil* entraînait la proposition de *les yeux* et *l'œil*).
Il semble important de changer la méthode de parcours de l'arbre ainsi que l'unité minimale de correction. Cela demande de changer également les règles de correction. De plus la méthode utilisée doit permettre de travailler sur des arbres incomplets et d'utiliser des règles heuristiques de correction.

Deux principes de CORSYN (entrées lexicalement correctes et un arbre syntaxique complet sur la phrase) sont contradictoires avec le cahier de charge de CELINE qui veut pouvoir travailler localement de manière multi-niveaux avec coopération de ces niveaux en vue d'une résolution globale. De plus, nous voulons une correction adaptée au

rédacteur. Nous avons donc décidé de ne pas poursuivre le développement de CORSYN.

3.3 VORTEX ET VORTEXPLUS

3.3.1 VORTEX

Le système VORTEX (Pérennou 86) (Pérennou 90) corrige les fautes d'usage et les fautes typographiques en utilisant un modèle stochastique.

VORTEX utilise un dictionnaire spécialisé, construit de manière semi-automatique à partir de la base de données BDLEX dans laquelle les mots sont découpés en sous-séquences orthographiques qui ont chacune une valeur phonétique (par exemple, *orthodoxe* est segmenté en *o-r-th-o-d-o-xe*. Le lexique représente le vocabulaire sous la forme d'un arbre factorisant les préfixes communs.

La structure du lexique a permis d'étendre ce système à l'analyse morphologique et d'en faire un système combinant lemmatisation et correction lexicale.

La représentation des mots X appellée gpo (de *groupe à problémes ortographiques* puis de *graphique-phonétique-orthographique*) est commune à l'écrit et à l'oral ; elle contient les représentations des désinences traduisant les variables linguistiques pour les classes flexionnelles indiquées dans le lexique.

Des règles de réécriture permettent d'engendrer des formes erronées qui se prononcent de la même façon (telles que th→t ou o→ho).

D'autres règles simulent les erreurs caractéristiques de l'utilisation d'un clavier (comme l'omission ou la permutation)

A partir d'une forme textuelle inconnue, un modèle probabiliste permet d'extraire du dictionnaire les mots à partir desquels ces règles produisent la forme erronée. Ce système est ainsi capable de traiter les erreurs typographiques, phonographiques et quelques erreurs de segmentation (coupure et soudure)

Pérennou (Pérennou 90) Pécatte (Pécatte 92) ont proposé de personnaliser VORTEX en le rendant capable de s'adapter à un usager ou à une classe d'usagers, de manière à optimiser les temps de correction en s'intéressant d'abord aux fautes les plus probables.

Le système de trace de VORTEX a permis d'obtenir des statistiques (type de l'erreur, position dans le mot ou couple lettres-son mis en cause).

L'observation, que les propositions de correction sont souvent à rejeter lorsqu'on prend en compte le contexte, a amené ensuite l'idée d'associer un correcteur lexical et un analyseur de phrase (syntaxique ou sémantique ou les deux …).

3.3.2 VORTEXPLUS

Dans cette optique, VORTEXPLUS (Pécatte 92) est composé :

- d'un analyseur morphologique tolérant (extension du correcteur lexical VORTEX),
- d'un correcteur lexical,
- d'un analyseur syntaxique (réseaux de transition augmenté) autorisant la présence de mots inconnus (affectation d'une catégorie unique X, affectation d'un coût et rajout sur les arcs du réseau correspondants aux catégories de la possibilité d'accepter la catégorie X) muni de fonctionnalité de correction d'erreurs syntaxiques;
- d'un générateur de correction pour les formes fléchies.

Les différentes propositions de correction sont proposées à l'analyseur qui rejette celles qui ne sont pas syntaxiquement correctes.

Si en fin d'analyse, il reste plusieurs propositions alors le système les trie par ordre décroissant des probabilités.

Des procédures permettent d'isoler des constructions comme les chiffres, les chiffres romains et certains sigles.

3.3.3 CONCLUSION

Ces deux systèmes peuvent être classés parmi les systèmes mono-correcteur.

VORTEX s'est révélé performant pour les questions de vitesse de traitement (par exemple sur un vocabulaire de 100 mots, le système se révèle 6 fois plus performant que l'algorithme de Lowrance et Wagner). Les connaissances du système étant plus ou moins imbriquées les unes dans les autres, une modification dynamique du comportement est relativement difficile d'où une difficulté d'adaptation à la situation. Par contre, la méthode de décodage n'exclut pas la possibilité de plusieurs lexiques et permet donc d'ajouter à un lexique de base des lexiques propres aux domaines des applications.

Par ailleurs l'architecture entrave la possibilité d'exécuter simultanément plusieurs recherches sur un même mot.

VORTEXPLUS est particulièrement intéressant par rapport à notre recherche puisqu'il fait apparaître la désambiguïsation lexicale par les résultats de l'analyse syntaxique.

3.4 ECLAIR

ECLAIR est un système permettant l'analyse lexicale et le traitement des formes textuelles inconnues appelées *mots non-attendus*. Il a été développé au L.I.M.S.I., au sein de l'équipe de Gérard Sabah par Sabine Letellier (Letellier 93). A la date du début de ma thèse, ECLAIR constituait le plus récent travail dans le domaine de la correction d'erreurs, d'où son grand intérêt.

3.4.1 TRAITEMENT D'UN TEXTE ET CLASSIFICATION DES NON-ATTENDUS LEXICAUX

L'ensemble des traitements est géré par un superviseur, le ***pilote***.

Les étapes de l'analyse et du traitement d'un texte sont les suivantes:

- la délimitation, la reconnaissance et l'interprétation des mots,
- le traitement des non-attendus éventuels.

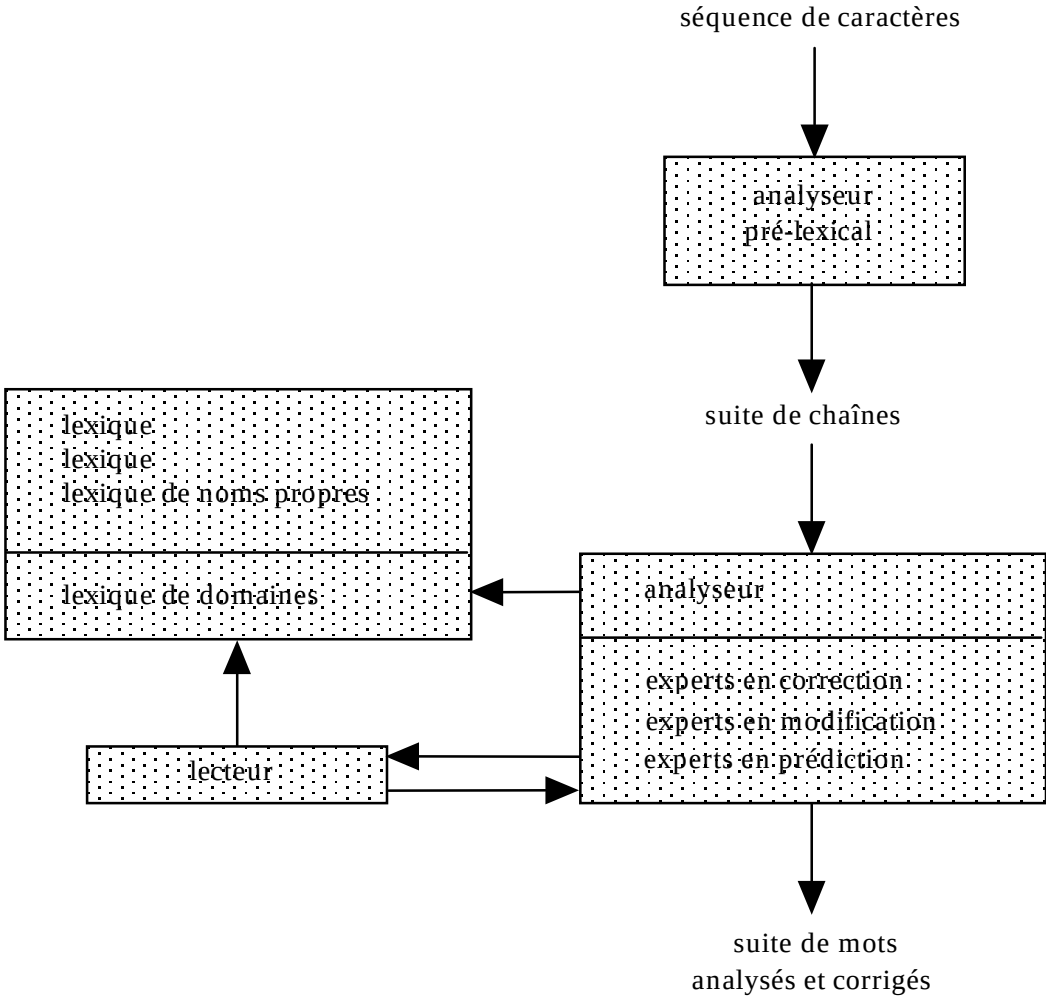


Figure 3.3 : Organisation des outils

Les classifications traditionnelles des non-attendus lexicaux ne satisfaisaient pas les concepteurs d'ECLAIR qui ont adopté la classification suivante :

- Erreurs typographiques :
  Elles consistent en ajout, suppression, substitution ou inversion de lettres.
- Erreurs phonographiques :
  Elles affectent peu ou n'affectent pas la prononciation du mot.
  Exemples : *pentaire*, *farmassi*, …

- Erreurs de segmentation :

Elles sont dues à la mauvaise utilisation d'un caractère séparateur (blanc, signe de ponctuation, trait d'union, apostrophe, …), sous la forme d'un ajout, d'une omission, d'une permutation ou d'une substitution.

Exemples : *diff iculté, cesproblèmes, catégoried es, renco,tre, …*

- Erreurs phonétiques :

Elles affectent "légèrement" les phonèmes d'un mot.
Exemples : *avian, chomin, …*

- Erreurs de construction :

Elles correspondent à la mauvaise construction de formes fléchies.
Exemples : *allerons, chevals, faisez, …*

3.4.2 CLASSES D'OUTILS

On appelle *module* ou *expert* un ensemble analyseur de mot et correcteur.
On appelle *outil* une association expert/lexique.

3.4.2.1 Les experts

Un certain nombre de classes d'**experts** ont été définies (voir figure 3.3) :

- La classe des *analyseurs pré-lexicaux*, qui prennent en charge la segmentation de la phrase, Ils lisent les caractères. Lors de la rencontre d'un caractère non alphabétique, ils effectuent une série de lectures jusqu'à rencontrer une nouvelle lettre. Le traitement dépend de la nature du ou des caractères non alphabétiques rencontrés.

- La classe des *analyseurs lexicaux*, qui vérifient l'existence des mots et recherchent les informations linguistiques à leur sujet. Ils sont divisés en plusieurs sous-classes :
  1. les analyseurs de mots alphabétiques,
  2. les analyseurs de nombres romains,
  3. les analyseurs de nombres arabes,
  4. les analyseurs de signes,
  5. les analyseurs d'abréviations,
  6. les analyseurs de formules,
  7. les analyseurs de symboles.

- La classe des *experts en traitement des non-attendus* qui s'occupent du traitement des erreurs et mots inconnus, elle-même divisée en plusieurs sous-classes :
  1. les experts en correction :
     - experts en correction alphabétique. Ils sont divisés en 8 sous-classes fonction du type des erreurs traitées, du nombre de paramètres requis, de la fréquence de lancement. Les "sous-classes" définies sont les suivantes :
       - correcteurs pour erreurs typographiques simples,
       - correcteurs pour erreurs typographiques complexes,
       - correcteurs pour erreurs de coupure,
       - correcteurs pour erreurs de soudure,

- correcteurs pour erreurs phonographiques,
- correcteurs pour erreurs phonétiques,
- correcteurs pour erreurs de construction,
- correcteurs pour combinaison d'erreurs.

- experts en correction portant sur des caractères non alphabétiques,

2. les experts en prédiction,
3. les experts en modification.

3.4.2.2 Les lexiques

Le système contient donc une base constituée de lexiques pour les mots d'usage courant et un ensemble de lexiques spécialisés chacun dans un domaine particulier. Les lexiques personnels des utilisateurs sont considérés comme des lexiques de domaines.
On a quatre classes de lexiques :
- les lexiques de mots communs,
- les lexiques de noms propres,
- les lexiques d'abréviations,
- les lexiques d'acronymes et de sigles.

Afin de déterminer un sous-ensemble de lexiques potentiels, le système demande en début de session à l'utilisateur de préciser si possible le(s) domaine(s) traité(s).

3.4.2.3 Les lecteurs

Les experts ne se chargent que du traitement du mot, et des modules spécialisés, appelés *lecteurs***,** se chargent des recherches dans le lexique.
Un même lecteur peut être sollicité par plusieurs experts.
La base d'outils est constituée de toutes les instances des différentes classes d'experts et de lexiques ainsi que des lecteurs.

3.4.2.4 Le pré-processeur

Il est responsable de la construction de la base d'instances (création et mise à jour) ce qui permet une plus grande flexibilité.
Il détermine les associations expert/lexique possibles et se charge de la récupération des informations "décrivant" les outils.
A l'introduction d'un nouvel expert ou lexique, il interroge l'utilisateur pour déterminer la classe de l'outil, le type du résultat de l'expert, le type des paramètres d'entrée des lecteurs et le type des lexiques lus par les lecteurs.
Si le type de résultat d'un expert coïncide avec le type des paramètres d'entrée d'un lecteur, alors l'expert est mis en association avec les lexiques lus par ce lecteur. Un expert n'a donc jamais besoin de connaître les lecteurs de la base, mais seulement les lexiques dont il a besoin.

3.4.3 COMBINAISONS D'ERREURS

Certains non-attendus peuvent être dus à des combinaisons d'erreurs.

> Exemple :
> *laconst itution* : erreur de soudure et erreur de coupure,
> *constitusionfrançaise* : erreur typographique et erreur de soudure,
> *super hiorité* : erreur de coupure et erreur phonographique.

Les combinaisons ne sont envisagées que lorsqu'un expert produit un résultat intermédiaire.

La possibilité de produire un résultat intermédiaire et la notion de résultat intermédiaire dépend de la classe de l'expert

- Pour les experts en modification : toute chaîne ayant subi une modification et n'appartenant pas au lexique est un résultat intermédiaire.
- Les experts en erreurs de coupure produisent des résultats intermédiaires, de même que les experts en erreurs de soudure (toute décomposition constituée d'au moins un mot appartenant au lexique).
- Les experts en erreurs de construction produisent des résultats intermédiaires du type : *cheuvals* donne *cheuvaux*.
- Les experts en erreurs non alphabétiques produisent également des résultats intermédiaires : *imagi,astion* donne *imaginastion*.
- Les experts en erreurs typographiques, phonographiques, et phonétiques, ne peuvent produire de résultats intermédiaires puisqu'ils recherchent dans le lexique des mots voisins.
- Les experts en prédiction ne peuvent fournir de résultats intermédiaires puisqu'ils ne produisent pas de mots.

Pour résoudre les combinaisons d'erreurs traitées par les experts ne produisant pas de résultats intermédiaires, le système a recours à des *experts en combinaisons d'erreurs*.

3.4.4 CONTRÔLE GLOBAL PAR UN PILOTE

3.4.4.1 Contrôle global

La solution retenue pour le contrôle est celui d'un contrôle global par un module indépendant : le *pilote*.

Dans le cas de combinaisons d'erreurs, les résultats intermédiaires sont alors pris en compte par le pilote.
La communication entre les experts et le pilote, ainsi que la communication entre les experts et les lecteurs, se fait par messages via un module intermédiaire appelé **exécuteur**.

Le pilote est capable de se charger des tâches suivantes :

- il observe et juge les experts, ce qui permet une meilleure sélection des outils ;
- il apprend à connaître les utilisateurs pour mieux répondre à leurs besoins ;
- il met à profit les temps morts pour mettre à jour des connaissances ou poursuivre des recherches.
- Le pilote est également chargé de résoudre les problèmes concernant la décomposition et la distribution des tâches ainsi que la synthèse des résultats.
- Le pilote gère l'analyse et la correction des mots dans différents contextes (nombre de processeurs et de lexiques variables, présence ou non d'un analyseur de phrases, etc…).

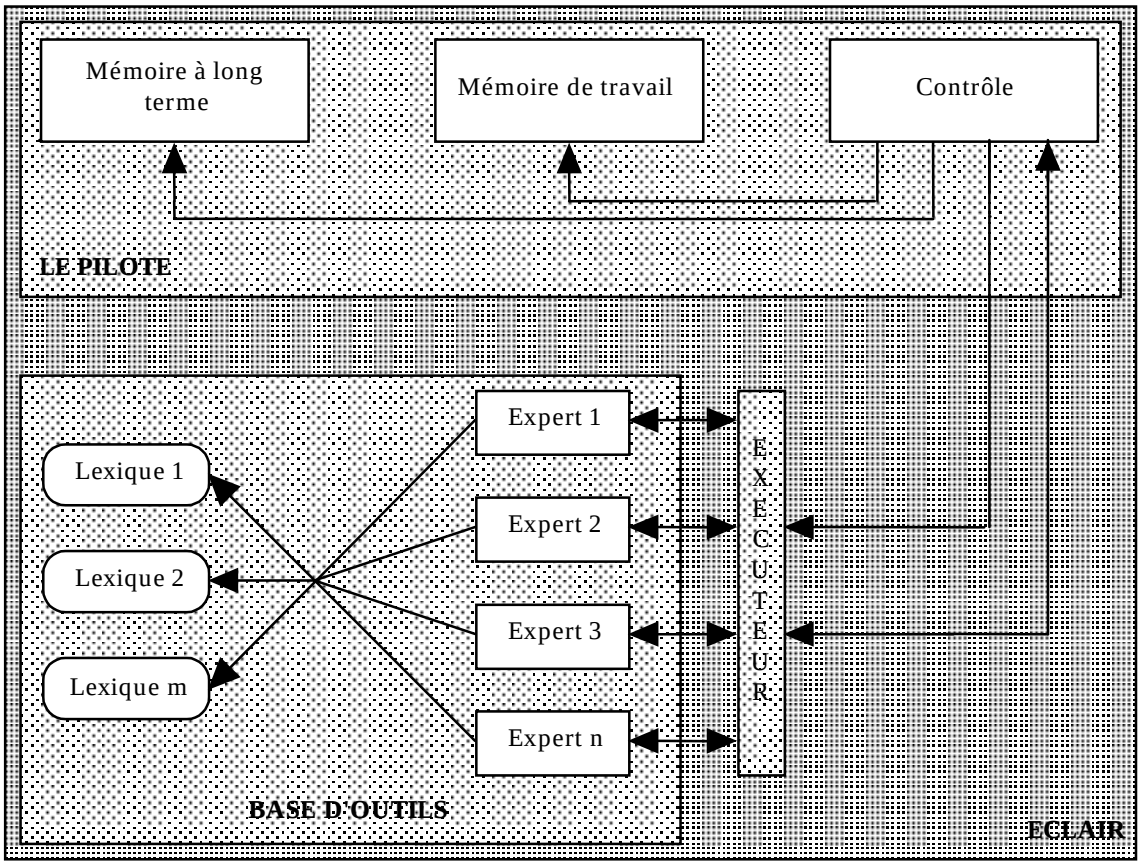


Figure 3.4 : Architecture générale du système

3.4.4.2 Connaissances du pilote

3.4.4.3 Les connaissances relatives aux outils

Nous distinguerons les connaissances sur les experts, celles sur les lexiques et celles sur les associations expert-lexique.

- Les connaissances sur les experts sont de deux types : statiques et dynamiques.
  - Les connaissances statiques sont :
    - l'identificateur de l'expert, dont le choix revient au concepteur de la base,
    - la classe ou sous-classe la plus basse dans la hiérarchie,
    - le nombre de paramètres (nombre de mots fournis en entrée),
    - le type des résultats, utile pour la détermination des associations expert/lexique, éventuellement accompagné d'informations sur les notes intervenant dans l'interclassement des propositions de corrections.
  - Les connaissances dynamiques sont :
    - la machine hôte de chaque expert ainsi que son état (actif ou attente),
    - le type du processus, le comportement, la spécialité et la nature du système au sein duquel s'insère éventuellement l'application.

- Les connaissances sur les lexiques :
  - L'identificateur du lexique (choisi par le concepteur de la base),
  - la taille du lexique : nombre de formes canoniques,
  - le type des mots : représentation habituelle ou phonétique,
  - la classe et le domaine d'appartenance du lexique pour orienter les sélections,
  - les caractéristiques des mots contenus : noms communs, verbes, noms propres, terminaisons particulières, etc…
  - L'ensemble des domaines connus du pilote est représenté sous la forme d'un réseau appelé *réseau des domaines et des lexiques*, dont chaque nœud est composé du nom d'un domaine, d'une liste des domaines qui lui sont rattachés, d'une liste de synonymes permettant de les désigner et de la liste des lexiques correspondants.

- Les connaissances sur les associations expert/lexique concernent :
  - le temps moyen d'exécution, qui est la somme du temps moyen d'exécution de la partie expert, du temps moyen de lecture dans le lexique et du temps moyen de transfert des données sur le réseau,
  - la position moyenne de la bonne proposition,
  - le nombre moyen de proposition émises.

Toutes ces connaissances sont recueillies par le pré-processeur.

3.4.4.4 Les connaissances relatives au contexte de travail

Elles concernent

- le profil de l'utilisateur. Les informations sont acquises par interrogation lors de la première utilisation puis par observation au cours des différentes sessions. Les informations recueillies lors de la première utilisation sont :
  - le type de l'utilisateur :
    - enfant (faible couverture lexicale),
    - non-francophone (faible couverture lexicale et confusion de phonèmes),
    - outil de saisie automatique (pas de fautes de construction ni d'orthographe mais erreurs typographiques), autre ;
  - la connaissance de la langue (ou couverture lexicale), de l'orthographe (ou capacité d'écrire correctement les mots que l'on connaît et pas seulement ceux que l'on a appris à écrire), et de la frappe (les fautes de frappe sont essentiellement d'ordre typographique et un faible niveau en frappe pourra orienter les recherches dans cette direction) ; pour chacune de ces trois informations, l'utilisateur s'attribue une note.

- le type d'application. Le type d'application permet au pilote d'adapter son comportement :
  - Le traitement de mots isolés se contente de vérifier l'existence d'un mot, et, le cas échéant, de proposer une correction. Ce type de traitement ne nécessite

jamais l'intervention d'un analyseur pré-lexical, ni celle d'un expert en prédiction ou même d'un expert à plusieurs paramètres.

- Les différentes formes de traitement de textes n'occasionnent pas tous le même type de fautes (la dictée doit faire rechercher essentiellement des fautes d'orthographe, dans la copie incertaine, le type de fautes est fonction des compétences de l'utilisateur, dans la copie correcte on rencontrera essentiellement des fautes de frappe, alors que l'improvisation devra orienter les recherches en fonction du type de fautes et des compétences de l'utilisateur).
- Le dialogue (généralement E.A.O.), contient plus de mots du domaine, et l'utilisateur, généralement occasionnel, commettra plus de fautes car son centre d'intérêt principal n'est pas l'orthographe.

- le domaine traité. Le(s) domaine(s) d'application(s) oriente(nt) le choix des lexiques.

- le type du texte. Le type du texte (table des matières, style télégraphique, compte-rendu technique, roman, article de journal,…) influencent le type de mots et l'analyseur.

- le public concerné. Le public concerné influence le choix des lexiques. En effet, un article de vulgarisation comportera peu, voire pas de mots spécialisés alors qu'une publication scientifique, par exemple, devra orienter vers des lexiques spécialisés.

- le type de clavier. Le type de clavier a une influence sur les fautes de frappe du fait des relations de voisinage entre les touches.
- les machines disponibles. Le système, étant capable de travailler dans un environnement avec un nombre de processeurs variables, doit donc connaître le nombre de machines disponibles pour faire la répartition des processus.

3.4.5 CONTRÔLE

3.4.5.1 Principes

Le pilote travaille sur les classes d'outils avant de raisonner sur les instances.

Les différentes étapes de son raisonnement, tantôt montant, tantôt descendant, sont à trois niveaux :

- le *niveau méta* : il constitue, avec le niveau résolution, la "partie dynamique" du traitement. Il assure la communication avec l'extérieur, détermine la stratégie de résolution et le plan statique adapté ;
- le *niveau diagnostic* : il constitue la "partie statique" du traitement. Il exécute le plan choisi pour sélectionner les associations classes d'experts/classes de lexiques les plus prometteuses ;
- le *niveau résolution* : il constitue, avec le niveau méta, la "partie dynamique" du traitement. Il instancie les associations sélectionnées, rend actifs les outils retenus et analyse les résultats.

Schémas de fonctionnement (voir figure 3.5) :

- La chaîne de caractères à traiter est fournie au niveau méta. Celui-ci, à l'aide des informations fournies par l'utilisateur et des éventuelles statistiques (quand il ne s'agit pas d'une première utilisation), sélectionne un plan statique pour la chaîne courante.
- Il passe alors la main au niveau diagnostic, qui choisit les associations classe d'expert/classe de lexique nécessaire au traitement de la chaîne.
- Le niveau diagnostic passe la main au niveau résolution, qui instancie les classes, et, via l'exécuteur, lance les associations expert/lexique sélectionnées.
- L'exécuteur renvoie au niveau résolution le résultats des experts (échec, mot reconnu ou proposition de correction).
- Si un résultat a été obtenu, il est retransmis au niveau diagnostic puis au niveau méta qui lance le traitement de la chaîne suivante.
- Si aucun résultat n'est obtenu, le niveau résolution continue l'exécution des instances. En cas d'épuisement des instances, le niveau résolution signale cet état de fait au niveau diagnostic qui va transmettre d'autres associations de classes, en se basant toujours sur le même plan sélectionné par le niveau méta.
- En cas d'échec de toutes les associations (erreur trop complexe, mot correct non reconnu, ou néologisme non détecté) ou de dépassement de temps (cas d'autant moins probable que le nombre de machines est élevé), le traitement est abandonné et l'abandon signalé au niveau méta.

Divers :

- Le pilote travaille à résoudre les problèmes lexicaux. Tous les types de non-attendus doivent être traités, y compris les combinaisons d'erreurs, les soudures de mots n'appartenant pas au même lexique, etc.
- Des principes d'analyse et de correction sont déterminés : pour chaque mot, il doit être fait une sélection de tous les outils pertinents, mais tous les experts ne doivent pas forcément être exécutés en association avec tous les lexiques. Cela dépendra des disponibilités matérielles.
- Une validation différée des résultats, permet de diminuer le risque d'échec sur des problèmes de segmentation, et d'avoir plus de chances de proposer la bonne solution.

  Exemple : *ana lyse* : avec une validation immédiate, le système proposerait *an* et *âne*. Avec une validation différée, le fait de trouver un deuxième non-attendu, entraînera le lancement d'un expert en erreurs de coupure, et la première proposition sera *analyse*. La validation est faite après traitement de deux mots (pour pouvoir traiter des non-attendus du type *anachronisme*) et en utilisant les caractéristiques des deux mots précédents (pour ne pas trop retarder les résultats tout en tenant compte du contexte). Les inconvénients sont la prise en charge de la reprise de correction si toutes les solutions proposées ont été refusées. Cette reprise doit être faite après traitement des deux mots suivants. Il faut donc conserver de nombreuses informations pour permettre au pilote de repartir de là où il s'était arrêté. On utilise à cet effet un **agenda** dont le fonctionnement sera expliqué plus loin.

3.4.5.2 Niveau diagnostic

Un plan est un traitement pour un type de chaînes, à l'aide d'un ensemble ordonné d'associations, suivant une stratégie bien précise.

Les traitements et les différents types de chaînes ne sont pas tous analysés et corrigés de la même façon. Pour une forme simple, on analysera (et éventuellement on corrigera) la chaîne complète. Pour une forme complexe, le traitement dépend de la nature des caractères intervenant et du nombre de chaînes impliquées.

Les caractéristiques d'une chaîne permettent d'orienter les recherches. Différents cas peuvent se présenter :

- les caractères la composant n'apportent pas d'informations utiles ;
- l'écriture des lettres (majuscule ou minuscule) peut permettre de distinguer les noms communs des noms propres, ou les acronymes, les sigles, etc. Cependant, il faut toujours tenir compte du fait que cette écriture peut être due à une erreur. Il faut également tenir compte de la fréquence d'apparition des formes (la présence de majuscules reflète plus souvent un nom propre qu'un nom commun mis en relief). Si toutes les lettres sont en majuscules, il s'agit probablement d'un nombre romain, d'un sigle, d'un acronyme, ou d'un mot d'une lettre commençant une phrase. Si on a une majuscule initiale : il peut s'agir du premier mot de la phrase, d'une acception particulière, d'une marque de déférence, ou … d'une erreur. S'il s'agit du premier mot de la phrase, ce peut être aussi bien un nom propre qu'un nom commun. Si tout est en minuscules (cas le plus fréquent), il s'agit généralement d'un nom commun, parfois d'une abréviation, ou d'une erreur (nom propre, sigle, …) Si le texte est entièrement en lettres capitales, la liste des associations traduit les statistiques d'emploi des différentes classes de lexiques. Mais les statistiques varient d'un texte à l'autre, ce qui donne plus de liberté au niveau résolution ;
- la taille peut apporter également beaucoup d'informations :
    - la plupart des petits mots (moins de quatre lettres), appartiennent au lexique général (exemple : *que, qui, moi, il, et, on, où, du, ces, la, son, coq, duc,…*),
    - Ils peuvent être aussi des abréviations, (exemple unité de mesure : *mn, km,…*repérée par la présence d'un nombre).
    - Ils peuvent également être un sigle, ou le début d'un sigle sans les points (exemple : *H L M, CV, UK, CIA,…* repérés parce qu'ils sont en majuscules). Cependant, on peut se trouver face à une erreur de coupure (exemple : *les ch iens*).
    - En revanche, les mots de plus de quinze caractères, sont généralement des noms propres ou des noms communs (pas de sigles, ni d'abréviations, très rarement des nombres romains). Mais on peut aussi avoir affaire à une erreur due à la pression prolongée d'une touche (exemple : *consssssssidérable*), ou une erreur de soudure (exemple : *chaqueentreprise*).

Les mots des formes composées avec apostrophes et/ou traits d'union, permettent aussi de diminuer l'espace de recherche, sachant que l'on a affaire soit à une forme union, soit à une forme liaison.

> Exemple du trait d'union
> - Quand le dernier mot appartient à la liste (*je, tu, il, elle, on, nous, vous, ils, elles, ce, lui, leur, ça, le, la, les, moi, toi, en*), c'est généralement une *forme liaison*. Sinon, c'est une *forme union*. Si on a une majuscule, on peut être face à un nom propre composé.
> - Si on a affaire à une forme union, on analyse la chaîne complète et les deux mots séparément.
> - Si on a affaire à une forme liaison, le premier mot est très probablement un verbe que l'on recherche donc dans un lexique général.
>
> Le type de raisonnement pour l'apostrophe est identique.

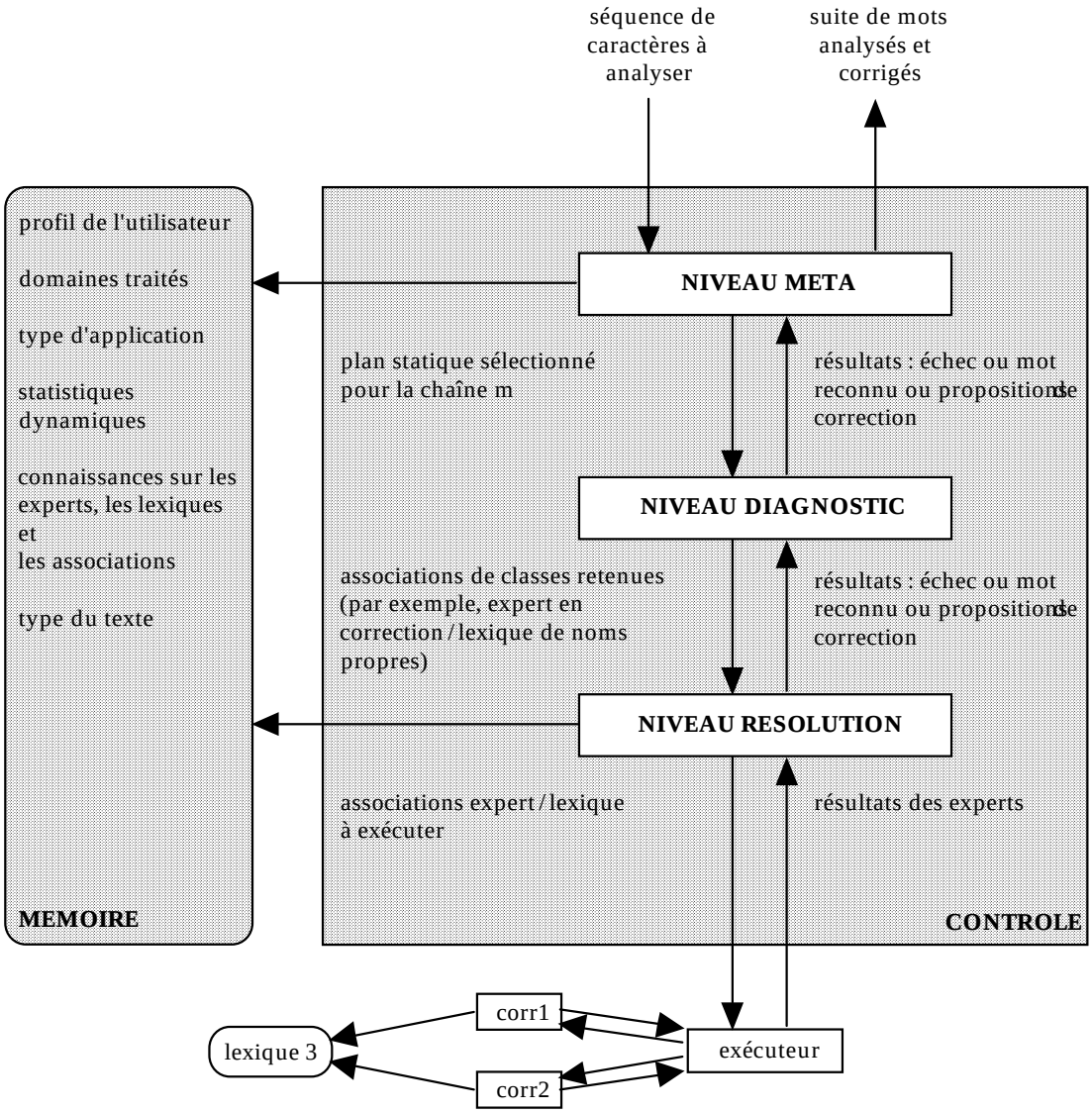


Figure 3.5 : Exemple de raisonnement

L'entourage d'une chaîne (son *contexte*) est également utile. ECLAIR exploite essentiellement le contexte gauche de la chaîne courante (un ou deux mots précédents). Face à un non-attendu, le contexte gauche peut apporter les indications suivantes :

- le mot précédent est également un non-attendu : on a fort probablement affaire à une erreur de segmentation ;
- les mots précédents sont un non-attendu et un signe : il s'agit probablement d'une faute frappe du type "," au lieu de "*n*" ;
- les mots précédents sont un non-attendu et un chiffre : il s'agit probablement d'une faute frappe du type "*4*" au lieu de "" ;
- les mots précédents sont un non-attendu et un symbole : il s'agit probablement d'une faute de frappe du type "%" au lieu de "*M*" ;
- le mot précédent est un nombre romain : le mot courant est en général "*ième*";
- le mot précédent est un nombre arabe : le mot courant est en général une unité de mesure.

Le contexte droit n'est exploité qu'au niveau de l'analyse pré-lexicale. Le niveau résolution le teste parfois pour ne pas lancer certains experts (exemple : lancement d'un expert en erreurs de coupure si le mot suivant est un signe).

Les stratégies de résolution dépendent de trois types de tests :

- tests relatifs au choix d'un lexique (orientés par la présence ou l'absence de majuscule(s)) appelés tests-MAJUS ;
- tests relatifs au choix d'un expert en traitement des erreurs (orientés par la taille et le contexte des chaînes) appelés tests-ERR ;
- tests relatifs à la composition des formes composées (qui n'ont en fait pas d'influence sur la stratégie).

Le choix de la stratégie est fonction du nombre de fautes faites par l'utilisateur (qui entraînera la suppression ou le retardement des tests-ERR), et du nombre de majuscules utilisées dans le texte (qui entraînera la suppression ou le retardement des tests-MAJUS).

On a donc ainsi quatre cas :

Quantité de mots commençant par une majuscule	Quantité de fautes	tests-MAJUS	tests-ERR
beaucoup	beaucoup	immédiats	immédiats
beaucoup	peu	immédiats	retardés
peu	peu	retardés	retardés
peu	beaucoup	immédiats	immédiats

Il existe des plans spéciaux pour les textes écrits entièrement en lettres capitales, textes pour lesquels les tests-MAJUS n'ont pas lieu d'être.

Les tests-ERR permettent l'obtention d'une liste ordonnée, mais le niveau résolution peut utiliser des connaissances exprimant le fait que les résultats des experts en erreurs de segmentations sont prioritaires sur les autres. On peut donc avoir les tests-ERR absents.
Suivant le nombre de tests réalisés et leur moment d'intervention, on a donc un raisonnement à profondeur variable.

On obtient ainsi les stratégies suivantes :

	tests-MAJUS	tests-ERR	Raisons du choix de la stratégie
	immédiats	absents	beaucoup de mots appartenant à des classes de lexiques différentes
beaucoup de machines	absents	absents	textes tout en majuscules
	retardés	absents	forte majorité de mots communs
	immédiats	immédiats	beaucoup d'erreurs et beaucoup ou peu de mots appartenant à des classes de lexiques différentes
	retardés	retardés	peu d'erreurs et grande majorité de mots communs

une ou quelques machines	immédiats	retardés	peu d'erreurs et beaucoup de mots appartenant à des classes de lexiques différentes
	absents	retardés	peu d'erreurs dans un texte tout en majuscules
	absents	immédiats	beaucoup d'erreurs dans un texte tout en majuscules

Seuls les types alphabétiques et formes composées les plus courantes sont concernés par ces stratégies. Les autres types ne sont traités que par un plan (ils existent 57 plans écrits).

Dès que l'exécution du plan est arrêtée (après succès ou abandon), le pilote met à jour la représentation lexicale de la phrase.

Tant qu'on n'a pas trouvé de solution ou abandonné les recherches (du fait de l'utilisateur ou de l'épuisement des outils), on fait des allers-retours entre le niveau diagnostic et le niveau résolution.

Quand le nombre de processeurs est inférieur au nombre d'associations, il faut établir un ordre d'exécution. Le niveau diagnostic établit un ordre sur les associations. Le niveau résolution classera ensuite les instances en fonction de cet ordre.

Le niveau diagnostic transmet l'ensemble des associations et non une partie, puis la suivante si échec.

Certaines associations, très peu probables, sont placées dans une sous-liste "dernier recours".

L'ordre transmis par le niveau diagnostic est partiel. La seule obligation est de constater la non-présence de la chaîne dans les lexiques avant d'entreprendre une correction. L'ordonnancement total est fait par le niveau résolution.

3.4.5.3 Niveau méta

Il choisit une stratégie de résolution en début de session puis teste constamment l'adéquation de celle-ci et en sélectionne une autre le cas échéant. Il est donc le garant de la flexibilité du comportement du système.

Il intervient tout au long de la session :

- Il est chargé de l'initialisation :
    - interrogation de l'utilisateur pour le recueil des informations,
    - choix de la stratégie de résolution et des outils,
    - affectation des notes initiales des associations expert/lexique ;
- En cours de session, il se charge :
    - de la communication avec l'utilisateur,
    - du déclenchement de l'expert d'analyse pré-lexicale,
    - de la modification éventuelle de la stratégie, de la gestion des statistiques dynamiques ;
- en fin de session, il sauvegarde les statistiques établies à des fins d'utilisation ultérieure.

La sélection et l'installation des outils utilisables est faite à partir des informations données par l'utilisateur :

- le choix des experts en fonction du type d'application,
- le choix des lexiques en fonction du type d'utilisateur et des domaines.

L'installation (répartition des outils sur les différents processeurs) est définitive car elle est très coûteuse en temps.

La gestion des taux d'utilisation des associations est faite à partir de la note initiale qui représente le taux d'utilisation prévu. Puis les notes issues des statistiques dynamiques (qui représentent le taux d'utilisation réellement observé) prennent le relai. Ces notes servent au tri des associations par le niveau résolution.

L'affectation des notes initiales est faite de façon variable selon les cas :

- l'utilisateur est inconnu : on utilise des règles de déduction de comportement d'après les informations recueillies auprès de l'utilisateur (type d'utilisateur, connaissance de l'orthographe, de la langue, de la frappe, type d'application). La note affectée à chaque expert est un pourcentage. La note affectée à chaque classe d'expert est affectée à toutes les associations contenant un expert de cette classe ;
- l'utilisateur est connu : le pilote a conservé les statistiques dynamiques de sessions précédentes. Elles servent de notes initiales si les conditions sont identiques ou proches (une dictée est équivalente à une recopie, une improvisation à un dialogue).

Le relais est ensuite pris par les statistiques dynamiques. En effet, les notes initiales, utiles au début, sont sujettes à une mauvaise auto-évaluation de l'utilisateur. Quand une instance rencontre un succès, on incrémente un compteur de succès des instances. Si plusieurs associations ont proposé la bonne solution, on incrémente un seul compteur, avec priorité à l'analyseur lexical.

Le calcul des notes est ensuite fondé sur le nombre de succès de l'association et le nombre de succès de son lexique.

La note de l'association correspond au nombre de succès de l'association multiplié par le nombre de succès du lexique et divisé par le nombre de mots traités.

Plus la note est élevée, meilleure elle est. Cependant on exécute toujours en premier l'association avec l'analyseur lexical pour vérifier que le mot est bien absent du lexique.

Les statistiques dynamiques sont utilisées à partir de 150 mots traités.

Le niveau méta se charge également du choix de la stratégie de résolution initiale. Il est fonction :

- du rapport nombre de machines disponibles / nombre d'experts,
- du type de texte,
- de la proportion de mots incorrects,
- de la quantité de mots avec majuscules.

A chaque combinaison de valeurs pouvant être prises par ces éléments correspond une stratégie. On a sept règles de sélection de stratégie.

Exemple :

- si texte entièrement écrit en lettres capitales et si beaucoup de machines alors choisir tests-MAJUS absents et tests-ERR absents ;
- si texte non entièrement écrits en lettres capitales et si pas beaucoup de machines et si utilisateur pas fortement sujet aux erreurs et si beaucoup de mots avec majuscules alors choisir tests-MAJUS immédiats et tests-ERR retardés.

La stratégie de résolution peut être remise en cause si elle s'avère inadaptée au texte courant.
Le pilote mémorise à cet effet 2 compteurs :

- le nombre de mots incorrects rencontrés depuis le début de la session (s'il est supérieur à 1/5 le pilote opte pour une stratégie avec tests-ERR immédiats ; sinon il opte pour une stratégie avec tests-ERR retardés) ;
- le nombre de mots analysés appartenant aux types des chaînes pouvant être corrigées.

Il mémorise également une liste contenant la classe du lexique d'appartenance des dix derniers mots analysés.

Si le nombre de mots commençant par une majuscule est supérieur ou égal à 1/10, il sélectionnera une stratégie avec tests-MAJUS immédiats, sinon il sélectionnera une stratégie avec tests-MAJUS retardés.

Il y a donc un réajustement permanent de la stratégie.

Des reprises de correction interviennent si l'utilisateur rejette toute les propositions de correction qui lui sont soumises. Le système doit alors reprendre le traitement là où il s'était arrêté.

A cet effet, on doit mémoriser :

- l'identificateur du plan statique ayant servi à traiter la chaîne,
- le point de reprise dans le plan,
- les propositions déjà rejetées,
- les propositions non encore testées,
- les instances non encore exécutées (mémorisées dans l'agenda).

Mécanisme de reprise :

- le niveau méta transmet au niveau diagnostic l'identificateur du plan à poursuivre, le point de reprise et l'agenda ;
- si l'agenda n'est pas vide, le niveau diagnostic communique au niveau résolution les associations contenues jusqu'à l'obtention d'un résultat satisfaisant, ou jusqu'à la poursuite du plan ;
- si l'agenda est vide, le traitement reprend au point de reprise.

Le niveau méta est également chargé de la communication avec l'extérieur (c'est-à-dire l'utilisateur). L'utilisateur peut :

- soumettre une suite de caractères ;

- rejeter une série de propositions de corrections en poursuivant ou non les recherches ;
- choisir une proposition de correction qui incrémente le nombre de succès de l'association qui l'a obtenue.

Lors de la terminaison d'une session, le niveau méta :

- sauvegarde les statistiques dynamiques pour les intégrer à la tendance générale de l'utilisateur,
- sauvegarde les notes des experts sous forme de pourcentage, ainsi que le nombre total de mots incorrects traités, le nombre total de mots que contenait le texte, la proportion de mots incorrects.

S'il y a un gros écart par rapport à la tendance générale habituelle (dû à de brusques progrès de l'utilisateur), les résultats des sessions précédentes seront abandonnés dans le cas où les progrès persistent sur un certain nombre de sessions (l'utilisateur peut avoir été aidé pour une session particulière).

Les résultats des sessions ne pouvant être conservés indéfiniment, le choix a été fait de ne plus les sauvegarder dès lors qu'ils ne modifient plus la tendance générale.

3.4.5.4 Niveau résolution

Il est chargé de déterminer les instances possibles et de les ordonner selon des critères de classement pertinents, puis d'analyser les résultats et de décider de la poursuite éventuelle des exécutions (instances suivantes ou résultats partiels).

Il gère également le problème de la multiplicité des domaines donc du choix des instances de lexiques.

- Une première solution consiste à explorer systématiquement tous les lexiques. C'est impossible quand le nombre de processeurs est réduit.
- Une deuxième solution consiste à se fonder sur les fréquences d'utilisation des lexiques, le plus courant étant le lexique général.
- ECLAIR adapte en fait la solution au contexte :
    - exploration systématique si l'on dispose d'un grand nombre de processeurs,
    - prise de risques si on en a peu. Les risques sont variables selon le type de tests, d'où l'utilité de la connaissance du type d'application et du type d'utilisateur.
- Le système utilise également les statistiques dynamiques : si un lexique s'avère être très utilisé, il sera consulté plus facilement.

Le niveau résolution se charge également de l'instanciation des associations de classes et du tri des instances (en se fondant sur les statistiques d'utilisation). Les notes déterminées par le niveau méta vont permettre d'établir un classement des outils.

Il est en revanche évident que les associations analyseur lexical / lexique doivent être faites avant les associations expert en traitement des non-attendus / lexique.

Si on a peu de ressources matérielles, les associations à très faible note ne seront pas lancées. On ne choisira que celles correspondant le mieux au profil de l'utilisateur et au type de l'application.

Lorsque les mots d'un domaine ont des caractéristiques visibles et que la chaîne courante comporte ces caractéristiques, le lexique de ce domaine associé à un analyseur lexical est avancé en début de liste.

Quand les ressources matérielles sont vraiment faibles, on place en fin de liste les associations coûteuses en temps.

L'agenda contient toutes les associations à exécuter, dans l'ordre de la liste construite et triée. Il est rempli par le pilote qui procède ensuite aux exécutions.

Le cumul des temps requis pour chaque association ne doit pas dépasser un seuil qui a été fixé à 3 secondes. Si le temps requis est supérieur, les résultats seront traités ultérieurement. Dans le cas d'experts "sûrs", le temps d'exécution peut être dépassé de 0,5 seconde maximum.

Dès exécution, l'instance est effacée de l'agenda.

Pour le traitement des résultats, le pilote attend la terminaison du travail des outils lancés ne dépassant pas un seuil de 3 secondes ; les autres résultats seront traités ultérieurement. Cinq cas sont possibles :

- la chaîne était un non-attendu et aucune proposition n'a été trouvée : si l'agenda n'est pas vide et qu'il y a encore du temps disponible, on continue les recherches, sinon on rend la main au niveau diagnostic ;
- le mot a été reconnu correct ;
- la chaîne était un non-attendu et des propositions de correction ont été émises : dans ce cas on analyse et trie les propositions de correction ;
- la chaîne était un non-attendu et il y a eu émission de résultats partiels et éventuellement de propositions de correction : on cherche alors à traiter les résultats partiels ;
- le mot semble correct (expert en prédiction) mais il n'a été trouvé dans aucun lexique de la base : on passe alors au mot suivant ou on poursuit les recherches si tout n'a pas été exploré.

Le tri des propositions de correction est fait en deux étapes :

- un premier tri est fondé sur les notes données par les experts (il est valable pour les différentes propositions d'un même expert) ;
- puis on applique les règles d'inter-classement se fondant sur les statistiques de succès des différents experts dans le contexte donné.

Le traitement des résultats partiels est fonction de la sûreté des résultats. Il existe des résultats peu sûrs issus d'experts en erreurs de coupure, des résultats moyennement sûrs issus d'experts en erreurs de construction ou de soudure et des résultats sûrs issus d'experts en modification.

La question qui se pose quand on a des résultats intermédiaires est la suivante : faut-il les exploiter et si oui, lesquels ?

- on va traiter d'office les résultats sûrs ;
- on traite les résultats moyennement sûrs si aucune autre proposition de correction n'a été trouvée ou si, pour un expert en erreurs de soudure, la partie non reconnue commence par une majuscule ;

- les résultats peu sûrs ne sont traités que s'il n'y a vraiment pas d'autres résultats.

Il faut ensuite déterminer les associations qui vont exploiter ces résultats intermédiaires. Ceci est fait selon le même principe que le tri de la liste initiale des experts.

3.4.6 REPRÉSENTATION LEXICALE DE LA PHRASE

Le système utilise des données construites dynamiquement ou déterminées en début de session (mémoire de travail) et des données persistantes d'une session à l'autre (mémoire à long terme).

Les connaissances permanentes sont principalement relatives aux outils et aux utilisateurs. La mémoire de travail comprend la représentation lexicale de la phrase, c'est-à-dire pour chaque chaîne traitée :

- la version initiale de la chaîne,
- son type (ex : alpha),
- ses traits (si la chaîne est correcte),
- l'identificateur du plan ayant servi au raisonnement,
- le point de reprise,
- l'agenda,
- les propositions rejetées,
- les propositions non encore testées,
- le mot suivant.

On construit avec ces informations un graphe. (voir figure 3.6)

Exemple :

Il est le malh eureux perddent dans cette affaire.

(pour *Il est le malheureux perdant dans cette affaire.*).

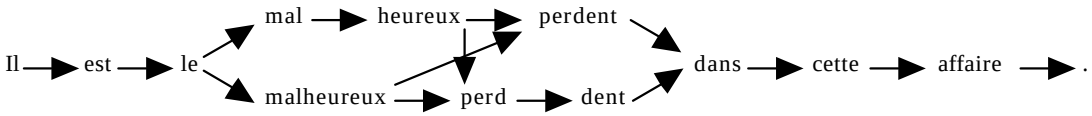


Figure 3.6 : graphe intermédiaire de reconnaissance de phrase[3]

3.4.7 MISE EN ŒUVRE DU SYSTÈME

La mise en œuvre du système peut se résumer ainsi :
Le pilote, transmet la séquence de caractères à un analyseur pré-lexical qui lui renvoie, après segmentation, les différents mots. Il retransmet ensuite chaque chaîne de caractères aux différents experts pour analyse et/ou correction, via l'exécuteur. (voir figure 3.7)

[3] Ce état est intermédiaire puisque *perdant* n'est pas encore proposé.

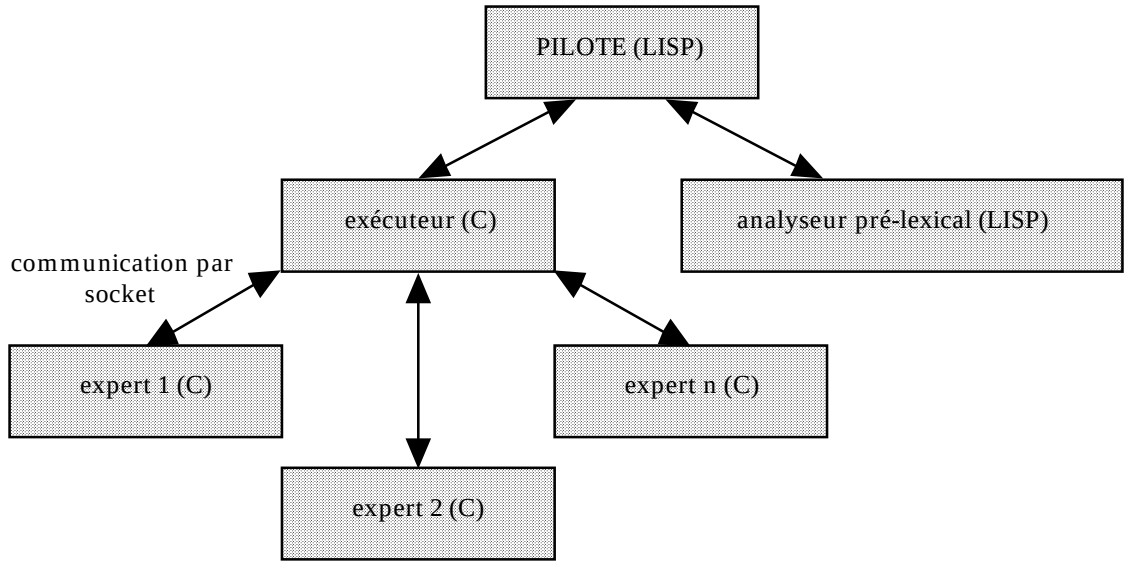


Figure 3.7 : Modules du système ECLAIR

3.4.8 ECLAIR ET CELINE

Différences fondamentales avec CELINE

- Par rapport à CELINE une grande différence existe : ECLAIR étant destiné à détecter et corriger les erreurs lexicales, ne traite ni les erreurs syntaxiques, ni les erreurs sémantiques. Il ne peut non plus détecter les erreurs cachées, puisqu'il ne fait pas d'analyse syntaxico-sémantique.

- La notion de contexte dans ECLAIR repose sur la catégorisation des formes adjacentes à la forme traitée en terme d'erreurs non-attendues, mot correct etc. Cela n'a rien à voir avec le contexte de CELINE qui est un contexte multi-niveaux.

- Le mode de fonctionnement global mono-niveau d'ECLAIR va donc être entièrement différent de celui multi-niveau de CELINE.

- La conséquence se retrouve dans les architectures : multi-experts pour ECLAIR et multi-agents pour CELINE.

- ECLAIR fait intervenir dans ses stratégies le nombre de machines disponibles et les temps de travail. Cet aspect n'est pas pris en compte par CELINE qui parie sur des progrès techniques rendant les temps de travaux raisonnables. De plus l'idée d'un très grand nombre de domaines avec les lexiques correspondants est traité au niveau d'un système réparti ou à la limite chaque lexique est sur un site et donc une machine différente.

Ressemblances de CELINE avec ECLAIR :

- ECLAIR est conçu pour traiter un maximum de formes correctes ou erronées ; c'est un système d'analyse et de correction lexicales traitant tous les types de mots quel que soit leur domaine d'appartenance, tous

les types de non-attendus, et ayant une forte autonomie. Cet aspect « universel » est également un des objectifs de CELINE.

- Pour obtenir une grande flexibilité, les concepteurs ont choisi une approche fortement modulaire. Il fallait donc intégrer différents analyseurs, correcteurs et des lexiques spécialisés dans le plus grand nombre de domaines.

3.5 CONCLUSION

Des quatre systèmes de correction des erreurs nous retiendrons concrètement pour être appliqués dans CELINE :

- de DECOR : les méthodes de corrections lexicales basées sur les clés ou la génération morphologique,
- de VORTEXPLUS : la correction lexicale validée par l'analyse syntaxique.
- d'ECLAIR : certains des éléments intervenant au niveau diagnostic et permettant de caractériser une forme textuelle comme par exemple les traitements du trait d'union ou de l'apostrophe.

Par ailleurs, même si les choix retenus pour CELINE vont être différents, ECLAIR amène une bonne réflexion sur le plan du fonctionnement général par ses trois niveaux méta, diagnostic et résolution.

4. MODÈLES HUMAINS EN CORRECTION DES ERREURS

Finalité du chapitre

Dans la terminologie des architectures multi-agents, les agents sont organisés en société, et par ses relations, chaque agent possède un comportement social qui peut tendre à reproduire le comportement humain. Dans cette optique, l'interprétation des productions d'un logiciel appelé *agent* sous la forme d'un comportement social peut ne pas être forcément très évidente. Si on peut comprendre facilement ces appellations, par exemple dans le cas de logiciels pilotant des robots nettoyeurs complètement autonomes dans des égouts (robots s'entraidant pour des tâches difficiles ou se donnant de l'énergie lorsque l'un d'entre eux se trouve batterie déchargée loin d'une source de branchements), par contre lorsqu'il s'agit d'un point d'une carte de géographie influant sur ses voisins, le vocabulaire semble assez loin de la réalité.

La question posée est donc : pour mériter le label *agent*, certains modules d'un système de correction d'erreurs peuvent-ils posséder un comportement social en reproduisant plus ou moins des modèles humains et si oui quel(s) modèle(s) ?

Notre réponse est positive et dans ce chapitre nous proposons deux premiers modèles humains pouvant sous-tendre les activités de correction des erreurs :

- Le modèle du maître d'école (agent dénommé *MAEC*) dans deux activités distinctes :
    1. enseignant, corrigeant l'orthographe de ses élèves considérés alors comme des rédacteurs et dont la connaissance des élèves augmente au fil des heures de classe.
    2. dirigeant une correction collective au tableau en sollicitant et coordonnant la coopération des élèves qui, dans cette circonstance, abandonnent le statut de rédacteur pour celui d'agent du système.
- Le modèle d'un décideur d'un service de renseignements (agent dénommé *DSR*), évaluant la validité des renseignements reçus en fonction de leur origine (source sûre, digne de foi etc.) et dont une des tâches est de classifier les sources disponibles.

Concrètement sur le plan de l'efficacité de la correction :

- L'implémentation du modèle humain du maître d'école « *enseignant* » aura pour finalité d'obtenir une connaissance précise de l'utilisateur, cette connaissance étant mise à disposition des agents. Cette connaissance va permettre d'être plus efficace (rapide) et plus sélectif (précis) dans la détection et la recherche des corrections.
- L'implémentation du modèle humain du directeur d'un service de renseignements aura pour finalité une connaissance des agents et plus particulièrement une quantification de leurs capacités et de leurs croyances. Cette quantification va permettre un calcul contribuant à une meilleure sélection des agents appelés à intervenir.
- La correction d'erreurs devant être la plus automatique possible, la combinaison des deux aspects ci-dessus contribue à une dispersion plus grande des probabilités des solutions concurrentes d'où une meilleure sélectivité.
- Le concept du maître d'école « *directeur de correction* » aura pour finalité de contribuer à clarifier le protocole d'actions et les échanges entre agents à l'occasion de leurs coopérations.

Résumé du chapitre
Dans une première partie, après un exemple illustrant les besoins de connaissances sur le rédacteur, nous justifierons l'appellation de maître d'école par une comparaison, dans la démarche de connaissances de l'élève, du comportement attendu du système vis à vis du rédacteur et du comportement d'un maître d'école vis à vis de ses élèves.

Dans une seconde partie, face à des agents imparfaits, nous définirons conceptuellement des coefficients permettant de chiffrer la confiance du système dans les productions de ces agents. Le directeur du service de renseignement va prendre en charge cette mission.

Le modèle du maître d'école est en quelque sorte un modèle historique vécu. Sur le plan de la correction d'erreurs nos premiers souvenirs viennent de l'école primaire (avec exercices au tableau). C'est là que nous avons appris comment rechercher des fautes et les corriger et l'idée d'un système informatique reproduisant les mêmes démarches est séduisante. Aussi dans une troisième partie, nous définirons le protocole général de fonctionnement de CELINE en le calquant sur cette activité humaine scolaire : nous allons rechercher un système dans lequel des spécialistes coopèrent de la même manière que des élèves faisant des travaux de groupe et dans lequel les connaissances peuvent s'exprimer, non par des formalismes mathématiques sophistiqués abstraits et très loin de la réalité, mais par des règles analogues à celles rencontrées dans les manuels scolaires (ou le Bescherelle ou bien encore le Grevisse).

4.1 POURQUOI UN MAÎTRE D'ÉCOLE ?

Reprenons la phrase nous ayant permis d'introduire la structure de données nécessaire :

je manje une lomme

comprenant deux fautes, l'une pouvant être considérée comme une faute phonétique et l'autre comme une faute de frappe (*l* voisin du *p* sur le clavier).
Nous allons successivement appliquer deux stratégies de résolution :

4.1.1 PREMIÈRE STRATÉGIE : COOPÉRATION DES AGENTS CLASSIQUES SUIVIE DE L'APPLICATION DE CONNAISSANCES SPÉCIFIQUES AU RÉDACTEUR.

- Rappel : le correcteur orthographique WORD 6 propose :

Forme textuelle 1	Forme textuelle 2	Forme textuelle 3	Forme textuelle 4
je	*mange* *manges* *manne* *manie* *mande* *manse* *mante*	*une*	*somme* *sommes* *comme* *gomme* *homme* *nomme* *pomme* *tomme*

Ces propositions lexicales conduisent donc à 7 * 8 = 56 phrases concurrentes pour la phrase initiale.

- Un module sémantique va autoriser les couples :

mange somme	manges somme	manie somme	mande somme
mange sommes	manges sommes	manie sommes	mande sommes
mange gomme	manges gomme	manie gomme	mande gomme
mange homme	manges homme	manie homme	mande homme
mange pomme	manges pomme	manie pomme	mande pomme
mange tomme	manges tomme	manie tomme	mande tomme

Un certain nombre d'autres couples vont pouvoir être éliminés par application des contraintes imposées par les agents coopérant à la résolution de la phrase :

1. Un analyseur lexical ou une modélisation de Markov des catégories morphologiques va imposer à la deuxième forme textuelle d'être un verbe.
2. Un analyseur lexical ou une modélisation de Markov des catégories morphologiques va imposer à la quatrième forme textuelle d'être un substantif.
3. Un module de vérification des accords va imposer à la deuxième forme textuelle d'être à la première personne.
4. Un module de vérification des accords va imposer à la quatrième forme textuelle d'être du genre *féminin* et du nombre *singulier*.

- Appliquons ces restrictions :

mange somme	~~manges~~ somme	manie somme	mande somme
mange ~~sommes~~	~~manges sommes~~	manie ~~sommes~~	mande ~~sommes~~
mange gomme	~~manges~~ gomme	manie gomme	mande gomme
mange ~~homme~~	~~manges homme~~	manie ~~homme~~	mande ~~homme~~
mange pomme	~~manges~~ pomme	manie pomme	mande pomme
mange tomme	~~manges~~ tomme	manie tomme	mande tomme

La coopération avec les agents classiques permet donc de réduire le nombre de solutions de 56 à 12.

Nous pouvons encore éliminer des solutions à condition de connaître notre rédacteur :

- Non-utilisation de tournure ou de mot précieux, nous n'obtenons alors plus que trois solutions sur les 56 propositions initiales :

  je mange une pomme
  je mange une gomme
  je mange une tomme

- Mauvaise utilisation du clavier : trois solutions résiduelles.

  je mange une pomme
  je manie une pomme
  je mande une pomme

- Non-utilisation de tournure ou de mot précieux et mauvaise utilisation du clavier : une seule solution résiduelle.

  je mange une pomme

Nous pouvons remarquer que, dans le scénario ci-dessus, l'obtention de la bonne solution est liée à l'application de règles difficiles à gérer comme la non-utilisation de tournures ou de mots précieux ou bien encore l'utilisation de la sémantique. Rappelons toutefois que le correcteur de WORD prend en compte certains éléments de style (par exemple pour *en quelque sorte* il préviendra *: N'abusez pas de cette expression vague et souvent inutile*).

4.1.2 DEUXIÈME STRATÉGIE : PRIORITÉ À L'UTILISATEUR :

Supposons maintenant que, dès le départ de l'analyse, nous appliquons nos connaissances sur l'utilisateur.

- Erreurs de transcriptions phonétiques fréquentes : *manje* est corrigé en *mange*
- Erreurs de frappe fréquentes : *lomme* est corrigé en *pomme*

En peu d'étapes nous obtenons la phrase corrigée *je mange une pomme*.

4.1.3 CONCLUSION DE CET EXEMPLE PARTICULIER

Récapitulons nos observations :

1. Les agents classiques pour tenter de résoudre la phrase ont dû s'appuyer sur des indices fragiles tel que le *je* demandant un verbe à la première personne ou le *une* imposant un genre *féminin* et un nombre *singulier* au mot suivant.
2. Les agents classiques ont été incapables de résoudre complètement la phrase (12 solutions résiduelles).
3. L'utilisation de connaissances spécifiques à l'utilisateur permet de réduire ce nombre de solutions à 3 ou même à une solution unique.
4. En termes de stratégie de résolution, l'utilisation des connaissances spécifiques à l'utilisateur permet de réduire fortement le degré d'ambiguïté, d'où une résolution plus rapide et plus sûre.
5. La connaissance du rédacteur permet d'aboutir à la solution sans faire intervenir la sémantique. Sans généraliser, on peut donc dire que parfois, la connaissance de l'utilisateur peut pallier les insuffisances ou imperfections du système.

Si la connaissance du rédacteur semble importante, il faut donc se donner les moyens d'obtenir cette connaissance. Un agent va donc être spécialement affecté à l'observation du rédacteur et va finalement décider des règles applicables pour ce rédacteur. Cet agent va être appelé maître d'école par analogie de comportement.

4.1.4 MODÈLE HUMAIN DU MAÎTRE D'ÉCOLE ENSEIGNANT

Nous pouvons dresser une comparaison entre le comportement de l'enseignant tout au long de sa démarche éducative et le comportement attendu de l'agent *MAEC*.

Plusieurs problèmes fondamentaux se posent :

- La définition intrinsèque du savoir linguistique d'un individu.
- L'évaluation de ce savoir (apprentissage puis suivi statistique).
- La codification des résultats de cette évaluation, sa mémorisation, sa validation, son suivi et sa mise à jour (par exemple : détermination statistique quantitative de coefficients de vraisemblance d'une règle pour les accords grammaticaux, matrices de transition de modèle de Markov d'ordre 1, 2 ou 3 directs ou inverses en ce qui concerne la désambiguïsation lexicale et le style).
- la détermination d'une liste d'heuristiques possibles,
- les stratégies à mettre en œuvre et le choix d'une heuristique particulière la plus pertinente possible.

Comparaison du comportement social du modèle humain et du modèle informatique	
Professeur	**Agents**
Premier stade : pré connaissance de l'élève ou du rédacteur	
	Définition standardisée des connaissances linguistiques d'un utilisateur (compétences lexicales : importance du vocabulaire, domaines; compétences grammaticales : jeu des règles grammaticales connues et appliquées, connues mais non appliquées, inconnues etc. ...). L'éducation nationale du premier et second degré nous fournit de multiples références dans le domaine (MEN 85) (MEN 97 a) MEN 97 b) (MEN 97 c). Le niveau de connaissances pourrait être codé de telle sorte que tout traitement de texte l'accepte dans son interface utilisateur (transmission par support physique sous forme de fichier « dossier linguistique »). Cette connaissance linguistique pourrait (à très long terme !) s'intégrer dans le cadre de l'interface Homme-Machine dans un ensemble beaucoup plus vaste de connaissance de l'homme par la machine (reconnaissance vocale, reconnaissance de caractères) mémorisée par exemple dans une carte à puces qui constituerait la carte d'identité informatique de l'individu.
Consultation du dossier scolaire de l'élève. Ce dossier scolaire contient une liste d'informations standardisées qu'instituteurs, professeurs, psychologues, médecins ont mis longtemps à définir.	
Dialogue direct avec l'élève pour définir son niveau, ses connaissances et méconnaissances, ses difficultés, ...	Un agent réactif passif d'initialisation va demander à un nouveau client de définir son niveau, ses connaissances et méconnaissances, ses difficultés en choisissant sur des listes pré-établies.
Connaissance initiale précise de l'élève / rédacteur	
Tests d'évaluation du niveau. Cette démarche est pratiquée en France sur le plan national au niveau du primaire, des sixièmes, des secondes,... Indépendamment des actions nationales, beaucoup d'enseignants font faire en début d'année des tests d'évaluation	L'agent d'initialisation peut proposer un ensemble de tests permettant de cerner le degré de compétence linguistique d'un utilisateur. Le multimédia nous donne même la possibilité de dictée... Cette étape permet une initialisation de divers paramètres (poids initiaux des règles, seuil d'application des règles, matrices de Markov, fautes phonétiques si dictée…).
Actions du maître et suivi de l'élève	
Dans le cadre d'une pédagogie différenciée, le maître va faire porter son effort sur les lacunes détectées ...	La stratégie générale et les stratégies particulières mises en œuvre par les agents de correction vont dépendre des évaluations précédentes.
Au fils des cours, des exercices, des dictées, l'enseignant va affiner sa connaissance de l'élève et va adapter la pédagogie mise en œuvre	Au fil des textes, les agents vont mettre à jour leurs connaissances de l'utilisateur (statistiques diverses, modélisation de Markov etc.) et vont faire évoluer leur comportement

Les connaissances et l'expérience du maître d'école comprennent

- des connaissances techniques précises (lexiques, règles de grammaire, ...),
- des connaissances obtenues empiriquement sur l'élève par son observation,
- des connaissances déductives : par exemple pour un utilisateur faisant de nombreuses fautes d'accord sur les substantifs et les adjectifs, on pourra suspecter légitimement qu'il fera également des fautes d'accord sur les verbes.

En sens inverse, un utilisateur ne faisant pas de fautes d'accord sur les verbes ne fera en général que peu de fautes d'accord sur les substantifs et les adjectifs. Par contre un élève ne faisant pas d'erreurs d'accord sur les substantifs et les adjectifs, pourra faire des erreurs sur les verbes qui constituent un autre domaine de connaissances.

4.1.5 LE MODÈLE LINGUISTIQUE PARTIEL SUFFISANT

Nous venons de voir que le maître d'école est un agent dont la finalité est de déterminer et gérer certaines connaissances relatives à un rédacteur. Cette connaissance s'obtient par l'observation du tableau des données lorsqu'une phrase complète est validée.

Le maître d'école va procéder par comparaison entre la phrase initiale et la phrase corrigée. Le maître d'école n'intervient jamais dans la structure. Il ne peut que la lire et l'interpréter. Par contre ses conclusions vont être distribuées aux agents concernés.

Les connaissances relatives à un rédacteur sont regroupées dans le modèle linguistique partiel suffisant (*MLPS*) de ce rédacteur que nous définirons complètement au chapitre 6.

A partir des phrases initiales ou corrigées du rédacteur et souvent par comparaison entre les deux, la tâche du MAEC consiste :

- A déterminer les *coefficients d'utilisation* par rédacteur des différents lexiques du système. Ces coefficients d'utilisation vont permettre de définir un ordonnancement des recherches dans les lexiques pour l'identification d'une forme textuelle. Le détail de la signification de ces coefficients et de leurs calculs sera présenté au chapitre 6 sur le MLPS.

  - A déterminer tous les éléments statistiques intervenant dans les modèles de Markov qui seront présentés au chapitre 5 et en particulier : détermination des *matrices de transitions* d'ordre un et d'ordre deux portant sur :
    1. les enchaînements des lettres dans les mots,
    2. les enchaînements des catégories morphologiques.

  - En attendant une bonne représentativité des modèles et une convergence des matrices, a déterminer une *grammaire de correction des modèles de Markov standardisés* (chapitre 6 sur le MPLS).

  - A déterminer les éléments intervenant dans la stratégie de correction d'erreurs en calculant un ensemble de taux tels que par exemple :

    1. Nature des fautes typographiques :
       Taux de substitution, d'omission …
       Taux de fautes de frappe quand la substitution d'un caractère se fait par celui d'une touche adjacente.

    2. Localisation des fautes :
       Taux de fautes portant sur la racine.
       Taux de fautes portant sur les désinences.

Taux de fautes sur les verbes.
Taux de fautes sur les substantifs.
Taux de fautes sur les adjectifs.
Taux de fautes sur les articles.

3. Fautes d'accord :

   Taux de fautes d'accord dans le groupe nominal.
   Taux de fautes d'accord dans le syntagme verbal.
   Taux de faute d'accord entre le groupe nominal et le groupe verbal.

4. Taux du nombre de fautes par mots (taux pour une faute, pour deux fautes, etc.)

5. Calcul de la *distance moyenne* entre chaque forme textuelle fausse et le mot choisi. On pourrait imaginer de nuancer cette distance moyenne par type de mots (substantifs, verbes …) ou bien par type de fautes.

6. Evaluation des règles du type : flemme du rédacteur. Sur un mot faux, taux du cas : nombre d'erreurs telles que la forme fausse comportait une marque du pluriel (x, s, …) et la forme textuelle finale retenue était bien au pluriel / nombre total de cas ou la forme fausse comportait une marque du pluriel (x, s, …)

Un taux d'erreurs sur la faute λ est de la forme τ_λ = (nombre de fautes du type λ) / nombre total de fautes. Bien entendu dans certain cas, la différence entre la forme fausse et la forme correcte peut être interprétée par des types d'erreurs différents. Dans ce cas on compte l'erreur pour tous les types d'erreurs.

Remarquons que d'autres types d'erreurs, et à peu près tous les types, pourraient profiter d'un suivi selon le même processus.

La comparaison des ordres de grandeur des taux va permettre de fixer la stratégie à utiliser : l'ordre d'application des traitements est l'ordre décroissant des taux c'est-à-dire que l'on cherche à traiter en premier les erreurs les plus fréquentes, les plus probables. Dans ce cas on travaille sur l'ensemble de la phrase, l'objectif étant de déterminer des îlots de confiance, zones « saines » de la phrase.

> Exemple :
> Pour la phrase *je manje une lomme*
> - pour un rédacteur dont le taux d'erreurs sur les articles est proche de zéro, le féminin de *une* ne va pas être mis en doute et le système va imposer un genre féminin à la forme remplaçante de *lomme*.
> - Pour un rédacteur, dont le taux d'erreurs sur les articles serait élevé, la validité de *une* va être mise en doute et le nombre de solutions résiduelles sur la phrase va augmenter.

Face à des solutions concurrentes pour une forme textuelle inconnue, les taux d'erreurs ou la distance de correction moyenne vont permettre aussi, dans certains cas, de faire un choix en retenant la plus vraisemblable : celle qui correspond au plus fort taux d'erreurs.

> Exemple
> *baeu* est corrigible en *beau* (permutation) ou en *bau* (faute d'insertion) si on suppose une seule faute par mot mais aussi en *beaux* ou en *bague* (permutation plus omission) si on accepte deux fautes par mots. Pour un rédacteur faisant des fautes de frappe mais peu de faute d'insertion et une faute par mot, nous avons des indices forts pour choisir la correction de *baeu* en *beau*.

Dans le chapitre 6 sur le modèle linguistique partiel suffisant nous compléterons ce paragraphe sur le maître d'école par son aspect dual au niveau de l'élève.

4.2 LE DÉCIDEUR DE S.R.

Postulat de base : les agents développés réellement sont imparfaits, on ne peut accepter sans méfiance leurs propositions. La confiance que le système peut placer dans un agent va être repérée par deux coefficients un *coefficient d'efficacité* et un *coefficient de crédibilité*. Les chapitres 6 et 9 sur le MLPS et CELINE nous permettront de compléter le processus de calcul.

L'agent DSR va être chargé de l'initialisation et du suivi des coefficients divers. Nous allons successivement considérer le secteur lexical et le secteur syntaxique.

4.2.1 COEFFICIENT D'EFFICACITÉ CE

4.2.1.1 Table de correspondance inter-agents

Ces coefficients sont relatifs aux capacités de l'agent par rapport au système CELINE. Nous nous plaçons cette fois à un autre point de vue que celui de la simple reconnaissance par un lexique : celui de la pertinence des traits délivrés en vue de l'analyse syntaxique et de la correction des fautes d'accord (par exemple : application de la méthode des structures). Cette évaluation est importante puisque le système se propose d'intégrer des agents lointains éventuellement imparfaits et dont les traits sont plus ou moins adaptés à ceux utilisés par CELINE qui va devoir utiliser une table de correspondance inter-traits. Il s'agit donc de quantifier la qualité de cette correspondance. Ce coefficient détermine l'adéquation entre le jeu de catégories morphologiques ou de super classes dans CELINE et dans l'agent en cause.

Nous formulerons quelques hypothèses pour le calcul de ces coefficients selon la correspondance entre les catégories et méta-classes utilisées par l'agent et celles utilisées par CELINE. Soit ca la cardinalité de l'ensemble Ea des catégories utilisées par l'agent et cc celle de l'ensemble Ec des catégories utilisées par CELINE. Nous supposerons que toutes les catégories de chacun des deux ensembles ont au moins un correspondant. Il se peut que certaines correspondances soient sémantiquement très limites mais c'est justement ce que l'on désire tester.

- Premier cas ca=cc (figure 4.1) :

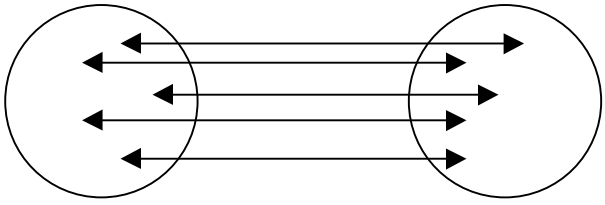


Figure 4.1

Correspondance biunivoque (1 pour 1) : les catégories sont identiques, on prend CE=1.

- Deuxième cas ca>cc (figure 4.2) :

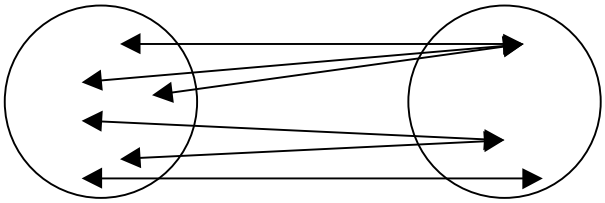


Figure 4.2

Tous les éléments de Ec ont au moins un correspondant dans Ea et tous les éléments de Ea ont au plus un correspondant dans Ec. On peut penser que dans ce cas, la description de l'agent est plus fine que celle de CELINE et on peut encore prendre CE = 1

- Troisième cas ca<cc (figure 4.3) :

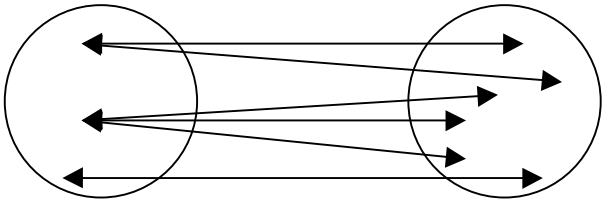


Figure 4.3

Dans ce cas CELINE possède une finesse de description plus grande et la mise en doute de l'efficacité du travail avec cet agent est plus sérieuse. Si par exemple à la catégorie *CAT* de l'agent peuvent correspondre *cat1 cat2 cat3* dans CELINE, le système va alors travailler en générant 3 formes textuelles avec respectivement *cat1, cat2 et cat3*. Nous voyons que dans ce cas, travailler avec l'agent revient à augmenter le degré d'ambiguïté et on comprend, que de ce fait, les performances générales soient moins bonnes. Cette fois le coefficient CE va devoir être calculé.

- Quatrième cas : cas général : mélange des deuxième et troisième cas.
  Comme dans le troisième cas, il va falloir déterminer le coefficient expérimentalement.

4.2.1.2 Définition du coefficient d'efficacité :

Coefficient d'efficacité = (Nombre de formes textuelles reconnues par l'agent et pour lesquelles l'analyse syntaxique et la vérification des accords ont abouti) / (Nombre total de formes textuelles reconnues par l'agent).

L'introduction d'un nouvel agent demande de définir la table de correspondance avec cet agent. A cette occasion on peut imaginer disposer de textes tests que l'on fait passer au nouvel agent de manière à initialiser les coefficients.

4.2.2 CRÉDIBILITÉ DES AGENTS

4.2.2.1 Coefficient de crédibilité des agents du secteur lexical

Rappelons que chaque agent lexical possède déjà un coefficient d'utilisation évalué par le MAEC (§ 4.1.5) et dont la détermination va être développée au chapitre 6 sur le M.L.P.S.).

Le coefficient de crédibilité va exprimer l'intérêt du système à utiliser les n lexiques i,j,k, ... en priorité ; l'utilisation de ces lexiques permettant d'optimiser tout à la fois la reconnaissance lexicale et la pertinence des informations en vue de la correction des erreurs par la méthode des structures.

Le coefficient de crédibilité va se calculer par le produit du coefficient d'utilisation et du coefficient d'efficacité : CC = CU * CE

Ce coefficient semble surtout intéressant en mode interactif. Il permet de diminuer le temps de réponse du système en évitant à l'utilisateur des temps d'attente trop longs.

4.2.2.2 Coefficient de crédibilité des agents du secteur syntaxique

Deux analyseurs syntaxiques travaillant sur la même phrase devraient fournir théoriquement les mêmes solutions. Cela n'est pas forcément le cas :

- comme nous l'avons vu les catégories morphologiques sont mises en correspondance par une table plus ou moins exacte.
- l'un peut avoir atteint un degré de développement plus avancé, posséder un codage plus riche de la grammaire etc.

On peut très bien obtenir une situation où la bonne solution est proposée tantôt par l'un, tantôt par l'autre, tantôt par les deux ou pas du tout proposée. De plus un analyseur peut être plus sélectif que l'autre. Il ne sert à rien d'avoir un analyseur qui propose toujours la bonne solution si cette solution n'est pas discernée parmi l'ensemble des solutions proposées. Dans le cadre d'un correcteur tendant à être automatique, il faut donc mettre ces agents en concurrence, ce qui implique de faire un choix entre leurs productions. Ce choix est nécessairement un choix non technique (c'est-à-dire ici non grammatical).

Nous proposons de déterminer ce coefficient à l'aide d'un corpus test. Dans ces conditions pour un nouvel agent d'analyse syntaxique :

> Coefficient de crédibilité de l'agent = (Nombre de phrases pour lesquelles l'analyse syntaxique par l'agent a abouti) / (Nombre total de phrases du corpus d'étude).

4.2.3 EXPÉRIENCE D'UN AGENT

L'idée générale des coefficients est de quantifier la confiance que l'on peut avoir dans la production d'un agent par rapport à une tâche donnée. Idéalement, le coefficient de crédibilité doit évoluer dynamiquement en fonction de la connaissance acquise par cet agent à travers son expérience, expérience qui doit le faire progresser en le rendant plus performant, plus fiable. Cette expérience peut en particulier se traduire par :

- une meilleure connaissance de l'agent humain,
- une meilleure connaissance des autres agents.

Considérons sur quelques exemples ce que représente « l'expérience » pour un agent.

1. Agent PSCM : l'application des modèles de Markov montre que, tant que les matrices de transitions n'ont pas atteint une certaine stabilité, cette modélisation est peu efficace. L'acquisition d'une stabilité des matrices demande des corpus importants (pouvant atteindre plusieurs centaines de milliers de mots). Dans le cas d'un correcteur orthographique, l'accumulation des travaux de correction correspond à un corpus de taille croissante. L'efficacité de cet agent va donc croître avec le temps puis se stabiliser.
2. L'agent MAEC va sans arrêt faire évoluer ses connaissances sur le rédacteur (lexiques utilisés, natures des fautes et taux rattaché …)
3. L'agent SDR va faire évoluer ses idées sur les agents du système (coefficient, règles heuristiques etc.)
4. L'agent TJRP est un agent de remplacement systématique d'une forme textuelle par une autre. Le système mis en place permet de déterminer, par choix de l'utilisateur ou automatiquement par comptage, les graphies qui doivent être remplacées systématiquement par d'autres. Il est bien évident qu'au fil des travaux cette connaissance va s'enrichir et évoluer.

4.2.4 INITIALISATION DES DIVERS COEFFICIENTS ET TAUX A L'AJOUT D'UN AGENT

A l'initialisation du système, deux solutions sont possibles :

1) Affectation arbitraire de valeurs.
   - Tous les coefficients sont mis à un et tous les agents ont alors le même coefficient.
   - Tous les taux d'erreurs sont mis à zéro et toutes les fautes ont la même probabilité.
   - Les coefficients reçoivent des valeurs par défaut déterminées plus ou moins « pifométriquement » par l'installateur en fonction de ce qu'il sait des performances de chaque agent.

2) Utilisation d'un corpus de test permettant une détermination rationnelle des divers paramètres (§ 5.2.2.2).

En cours de fonctionnement, les coefficients vont ensuite diverger selon l'efficacité des agents.

L'ajout d'un agent en cours de fonctionnement, va demander d'initialiser les différentes variables le concernant. Là encore, deux solutions sont possibles :

1) Affectation arbitraire de valeurs.
   - On ne peut plus choisir la valeur 1 pour un coefficient car cela donnerait a priori un trop grand pouvoir de décision à cet agent, agent qui n'a pas encore fait la preuve de son efficacité et ne possède pas d'expérience.
   - On ne peut plus choisir la valeur 0 pour un taux d'erreurs car cela entraînerait une non prise en compte de ce type d'erreur au cas où elle serait mise en concurrence avec d'autres erreurs.
   - Une solution est de leur affecter la moyenne des coefficients des agents ou la moyenne des taux d'erreurs. On peut aussi moduler « pifométriquement » cette moyenne.

2) Utilisation d'un corpus de test (§ 5.2.2.2).

4.2.5 CORRECTION INTERACTIVE ET VALIDATION DU MODE AUTOMATIQUE

Face à une phrase initiale avec ou sans erreurs, le système peut :

- échouer totalement,
- proposer une solution unique,
- proposer plusieurs solutions affectées de poids.

Tant que le système ne se juge pas autonome, il va demander l'aide de l'agent humain pour choisir entre les solutions résiduelles. De même que dans WORD, on peut choisir les affichages *Normal, Page, Lecture à l'écran, Plan, Document maître,* nous proposons un affichage *Correction* dont nous donnons un exemple (figure 4.4).

Phrase initiale	**N°**	**Propositions de phrases corrigées**	Poids
Je manje une lomme	01	Je mange une pomme	0.9200
	02	Je mange une gomme	0.4534
	03	Je mange une tomme	0.4234
	04	Je mande une pomme	0.2128
	05	Je mande une gomme	0.2001
	06	Je mande une tomme	0.1870
	07	Je manie une pomme	0.1600
	08	Je manie une gomme	0.1600
	09	Je manie une tomme	0.1600
	…	…	…
Corrigez la phrase initiale ou Double cliquez sur la correction sélectionnée			
Le système corrige correctement actuellement 84 phrases sur 100. **: Maintien du mode interactif de correction** : **Passage au mode automatique de correction**			

Figure 4.4
(Remarque : la colonne poids est une option d'affichage que le rédacteur peut choisir d'actionner)

Le problème du système est de savoir si la solution qu'il place en N°1 est choisie par le rédacteur comme étant la bonne (dans le cas de plusieurs solutions ex-aequo arrivant en tête, le système considère qu'il a échoué). Pour cela l'agent DSR va tenir à jour un taux d'auto-correction défini par :

> *Taux = (Nombre de fois où le rédacteur choisit la solution placée en tête par le système)/Nombre total de phrases traitées.*

Nous voyons que ce taux prend en compte les cas d'échec total sur les phrases ainsi que les cas où plusieurs solutions arrivent en tête ex-æquo.

Si le taux dépasse une certaine valeur seuil à choisir par le rédacteur, le système va pouvoir demander au rédacteur s'il désire le mode de correction automatique. Dans ce cas, l'affichage est modifié par l'ajout d'une ligne permettant à l'utilisateur de choisir ou non le mode automatique par une case à cliquer.

Le rédacteur peut, à tout moment, repasser en mode interactif.

De son côté par prudence, le système va scruter les valeurs des divers coefficients et taux qu'il continue par ailleurs à évaluer. Un changement brutal l'amène à remettre en doute sa capacité à l'auto-correction et il va prendre contact avec l'agent humain pour lui demander confirmation du maintien du mode automatique. Dans ce cas, il va afficher une fenêtre :

Anomalie possible du système de correction.
: Maintien du mode automatique de correction
: Passage au mode interactif de correction

Bien entendu, actuellement nous savons que le mode de correction automatique n'est pas généralisable compte tenu de l'état de l'art et de l'absence de sémantique. Nous pensons que, par contre, dans des conditions particulières, ce mode commence à être possible et que les principes vus peuvent être intégrés comme aide à la rédaction par exemple dans le cas d'applications scolaires (MEN 85).

4.3 PROTOCOLE DE CORRECTION COLLECTIVE DANS CELINE

Nous allons définir le protocole de correction dans Céline à travers un scénario-fiction dont nous allons préciser la distribution des rôles :

Éléments du système de correction	***Rôle dans le scénario***
Traitement de texte et rédacteur	Non représentés ; ils ont simplement produit une phrase à corriger.
Logiciel multi-agents de correction	Les agents comprennent le Maître d'école et les élèves d'une classe
	Le Maître d'école est chargé du contrôle et chaque élève agent du système intervient selon ses compétences

Nous avons vu au chapitre 2 que la coopération entre agents allait pouvoir se faire par l'intermédiaire d'un tableau structure de données. Nous pouvons considérer que cette structure est écrite au tableau et que, comme dans l'exemple traité, les élèves tour à tour rayent des mots, complètent d'autres informations etc. (figure 4.5). Nous nous approchons donc de la notion-métaphore de tableau noir utilisée dans les systèmes multi-agents (Nii 1986).

Le mode de fonctionnement peut se préciser :

- Le Maître d'école écrit la phrase initiale dans le tableau *structure de données* dont la définition a commencé au chapitre 2.

- Le DSR choisit parmi les élèves, les élèves agents lexicaux, les élèves agents d'analyse syntaxique, les élèves agents markoviens … lui semblant les plus aptes à résoudre le problème. Dans son choix interviennent les spécificités du rédacteur, du type de texte pour un rédacteur, de la partie dans un texte donné, etc. (voir chapitre 6 sur le MLPS).

- Les agents-élèves désignés vont braquer leur attention sur le tableau et selon leur domaine de compétence, ils vont refuser, proposer, remplacer ou marquer leur indifférence (incapacité).

- Il n'y a pas d'ordre, de stratégie prédéfinie : chacun à sa cadence, lorsqu'il est libre, amène sa bribe d'informations. On a bien alors une coopération atomique de tous.

- Lorsqu'un agent-élève trouve une information, sans autorisation, il se lève et va porter au tableau la marque de validité qu'il vient de déterminer ou le champ qu'il complète ou encore une nouvelle ligne.

- Lorsqu'un agent-élève refuse une ligne, il barre dans tout le tableau toutes les informations utilisant cette ligne.

- Si un agent (un élève) a besoin d'une information, il peut exceptionnellement la demander directement à un autre agent.

Le MAEC observe le tableau. Il tient le décompte des solutions partielles ou totales trouvées. Lorsque, plus rien ne change dans le tableau (c'est-à-dire : tous les agents-élèves ont donné leur avis sur tout ou bien une garde de délai a été dépassée) il fait le bilan.

Figure 4.5

- Si une solution unique a été trouvée, alors elle est considérée comme valide et elle est donc retenue pour être recopiée au propre (proposée à l'agent humain).

- Si plusieurs solutions ont été trouvées, le MAEC et le DSR vont se consulter pour leur affecter quantitativement un poids. Ils vont coopérer pour sélectionner une solution unique. Pour l'instant les poids sont de simples produits des coefficients de crédibilité, de probabilité, de taux d'erreurs.

> Exemple :
> Supposons qu'après une forme textuelle FT0 (catégorie morphologique Cat_0) connue se trouve une forme textuelle inconnue FTX à corriger.

> Le système propose deux formes substituantes concurrentes FTC1 (catégorie morphologique Cat_1) reconnue par le lexique i et FTC2 (catégorie morphologique Cat_2) reconnue par le lexique j.
> La comparaison de FTX et de FTC1 permet de déterminer le type d'erreurs et donc le taux τ_1 qui l'accompagne.
> La comparaison de FTX et de FTC2 permet de déterminer le type d'erreurs et donc le taux τ_2 qui l'accompagne.

Forme FTC_1	Forme FTC_2
Appartenance au lexique i : Coef CC_i	Appartenance au lexique j : Coef CC_j
Enchaînement des catégories morphologiques Cat_0 Cat_1 probabilité p_1	Enchaînement des catégories morphologiques Cat_0 Cat_1 probabilité p_2
Erreur de type k : taux τ_1	Erreur de type k : taux τ_2
Poids de la forme FTC_1 : $CC_i * p_1 * \tau_1$	Poids de la forme FTC_2 : $CC_i * p_2 * \tau_2$

- Si aucune solution n'a été trouvée, alors le MAEC peut appeler d'autres agents-élèves à la rescousse. On retourne alors à la case départ : tout recommence.

- Si toutes ressources essayées, il n'y a toujours pas de solution alors l'échec est signalé au rédacteur.

Avant d'effacer le tableau et de passer à l'exercice d'après (la phrase suivante), le MAEC et le DSR vont observer la phrase initiale et la ou les phrases retenues comme solution et mettre à jour leurs statistiques, le MAEC sur le rédacteur et le DSR sur les agents.

Une question se pose : est-ce que des changements dans l'ordre de réponse des agents peut amener une solution finale différente ?

La réponse est non. En effet, notre choix consiste à ne pas refuser de solution et à tenir compte de toutes les propositions (limitées à celles que permet la compétence du système). Toutes les alternatives vont donc être examinées et toutes les solutions trouvées.

Par contre, l'ordre d'intervention des agents va jouer sur le temps de travail global. En vue d'une optimisation de la recherche, il faudrait peut-être rechercher des stratégies différentes. Face à un rédacteur connu, certaines règles heuristiques pourront entraîner une amélioration du temps de correction.

Supposons par exemple qu'un rédacteur fasse peu de fautes de syntaxe mais beaucoup de fautes d'accord. Les modèles de Markov n'ont pas alors la même importance et peuvent passer derrière. L'analyseur syntaxique pourra travailler dès la fin des analyses lexicales et, à partir de là, dès que les premières structures seront inscrites au tableau les agents des accords pourront s'atteler à la tâche.

Par ailleurs, une pré-étude semble montrer le phénomène d'auto-optimisation mentionné au chapitre 2 (§2.3.5.3).

4.4 CONCLUSION

La finalité de ce chapitre était d'établir le comportement social de deux des agents du système CELINE. Nous venons de présenter un certain nombre d'éléments sur le maître d'école et le DSR.

Nous avons montré que la connaissance du rédacteur était essentielle à la correction des erreurs. La tâche principale de l'agent Maître d'école va donc consister, après un apprentissage du rédacteur, à fournir au système les éléments qui permettront une optimisation des prises de décision.

De la même façon pour le DSR, nous avons défini quel devait être son observation du système. Cet agent interviendra sur le système en faisant abstraction du rédacteur.

Nous avons présenté des agents peu typés et évolutifs. En effet, rappelons que le système CELINE, en principe destiné à la corrections des erreurs, se présente dans la réalité sous la forme d'un ensemble d'agents, formant une boite à outils. D'autres utilisations sont facilement envisageables telle que par exemple des applications pédagogiques dans le domaine de l'orthographe ou de l'aide à la rédaction.

En outre ce chapitre a permis de compléter les modalités du contrôle dans CELINE. Le chapitre 2 avait permit d'établir la structure de données, futur tableau noir du système multi-agents. Ce chapitre 4 vient de préciser le mode d'intervention des agents sur ce tableau noir.

Une partie de l'apprentissage va s'obtenir par une observation statistique du fonctionnement. Avant d'approfondir certains aspects au chapitre 6, nous allons mettre en place les modélisations de Markov , modélisations qui vont être fondamentales dans la connaissance d'un rédacteur.

5. MODÉLISATION DE MARKOV

Finalité du chapitre
Les modèles de Markov occupent un chapitre entier pour deux raisons bien distinctes :

- ils correspondent à une méthodologie d'exploration et de définition d'une langue naturelle vivante,
- ils participent activement au fonctionnement de CELINE.

Une langue vivante évolue et si nous considérons en langue française, du Rabelais, une page de l'Académie française ou du rap, chacun de ces textes représente sûrement une étape dans l'évolution de notre langue. Celle-ci évolue lentement mais continuellement, sans qu'à une date t, on puisse définir son état avec précision. Son évolution réelle ne peut pas être résumée dans les mises à jour périodiques de nos académiciens. D'une certaine façon, ceux-ci définissent la langue officielle codifiée par une prise en compte (prudente !) des habitudes de parole. Pour sélectionner de nouvelles habitudes de parole, ils opèrent en quelque sorte qualitativement. Par rapport à cette façon de faire, les modèles de Markov peuvent être considérés comme un processus de même nature mais sans doute plus rationnel car quantitativement basé sur des statistiques portant sur les productions des rédacteurs.

Qu'un linguiste rejette certains formalismes trop mathématiques des informaticiens me semble un fait acceptable ; ces formalismes ne correspondent pas à quelque chose de physique par rapport à l'apprentissage d'une langue. En particulier, cette thèse ne contient aucun développement sur des grammaires X, Y ou Z ... Ces grammaires représentent quelques règles générales intéressantes pour un domaine donné mais le grand nombre de domaines et, dans un domaine, le grand nombre d'exceptions rendent leurs implantations informatiques difficiles.

Une idée pédagogique courante est que l'apprentissage d'une langue s'effectue en extension. Nous considérerons donc une espèce d'équivalence entre les modèles de Markov appliqués à deux niveaux (les mots comme formes apparentes et les catégories morphologiques comme formes cachées) et la grammaire générale virtuelle supposée que personne n'arrive à codifier. Le procédé va remplacer une impossibilité d'écrire des règles générales par la difficulté d'obtenir un corpus de taille suffisante.

Résumé du chapitre
Dans une première partie nous présenterons les modèles de Markov et nous expliciterons l'utilisation quantifiée possible de ces modèles. Dans une seconde partie nous résumerons les principaux éléments statistiques nécessaires au calcul des modèles probabilistes. Une troisième partie permettra de présenter les théories et les algorithmes de mise en œuvre de filtres Markoviens.
Pour terminer, en se basant sur les résultats d'une expérimentation donnée en annexe, nous tirerons des conclusions pratiques d'emplois et nous déterminerons les règles retenues dans CELINE pour l'utilisation des modèles. Nous y verrons notamment comment CELINE peut changer de stratégie en fonction du rédacteur et d'une situation particulière. Nous ne serons plus alors très loin de la mise en œuvre de la coopération entre agents.

Les exemples donnés dans ce chapitre seront tous relatifs au domaine de la désambiguïsation des catégories morphologiques. Mais nous avons déjà signalé que CELINE utilise également des modèles de Markov dans d'autres domaines comme par

exemple la détermination de la première lettre d'une forme textuelle inconnue ou encore l'ordre des lettres à essayer.

5.1 PROBLÉMATIQUE DE L'UTILISATION DES MODÈLES DANS CELINE

Notre problématique (§ 2.3) va demander, dans certains cas, un comportement pouvant être légèrement différent de celui pratiqué dans d'autres domaines utilisant fréquemment les modèles de Markov. Nous pouvons envisager des textes saisis au clavier, des textes lus par des lecteurs optiques ou des textes issus de la reconnaissance de la parole. Les choix retenus vont, bien entendu, être fonction de cette spécificité (Perennou 86).

Par exemple :

1. Comme nous l'avons vu au chapitre 1, la correction demande de réunir un ensemble d'indices, par exemple proposer une catégorie morphologique pour le ou les mots inconnus. L'objectif est de compléter la structure de données présentée au § 2.3.5, structure de données que nous retrouverons au chapitre 8 (§ 8.7) sous la forme d'un tableau noir. Si nous considérons une phrase entachée d'erreurs comme un vecteur de mots, nos efforts portent sur le ou les mots à corriger. Les différents éléments du vecteur n'ont pas la même importance et le même rôle. Les mots reconnus vont constituer des îlots de confiance à partir desquels nous allons chercher à obtenir des renseignements sur les mots non reconnus correspondants à des mots inconnus des lexiques pris en compte ou à des mots entachés d'erreurs. Nous travaillons localement et non globalement.
2. Notre connaissance du rédacteur est insuffisante. Les matrices de transitions vont être imprécises, non stables et évolutives au fur et à mesure que notre connaissance du rédacteur augmente.

- Notre agent principal utilisant les modèles de Markov s'appelle PSCM c'est-à-dire Proposition Statistique de Catégories Morphologiques. Son rôle premier va être de proposer une catégorie morphologique pour le mot inconnu ou suspect (n°i) en connaissant la catégorie morphologique du mot i-1 (modèle d'ordre un) ou les catégories morphologiques des mots n° i-2 et n° i-1 (modèle d'ordre 2).

- Nous disposons également d'un agent Pilmark dont l'objectif est de faire de la désambiguïsation globale. Cet agent ne peut-être utilisé que dans le cas de corpus important permettant une bonne connaissance du rédacteur. Dans ce cas notre problématique, par rapport au modèle de Markov, rejoint celle de la RAP.

Notre présentation des modèles de Markov va tenir compte de cette spécificité, l'objectif étant de présenter les notions utiles et non l'état de l'art des modèles de Markov.

5.2 MODÈLES DE MARKOV

5.2.1 INTRODUCTION AUX MODÈLES DE MARKOV

5.2.1.1 Processus d'émission ou processus externe

Considérons la phrase :

Il faudrait la remonter

Nous pouvons considérer chacun des mots de cette phrase comme émis par un processus appelé *processus d'émission*.

La chaîne totale appelée *chaîne externe* Y_T = *Il faudrait la remonter* va correspondre à l'ensemble des 4 mots :

y_1 : *il*
y_2 : *faudrait*
y_3 : *la*
y_4 : *remonter*

L'élément générique est y_t avec t compris entre 1 et T=4

On peut considérer des sous-chaînes de la chaîne totale :

Y_1 = *Il*
Y_2 = *Il faudrait*
Y_3 = *Il faudrait la*
Y_4 = *Il faudrait la remonter*

5.2.1.2 Processus de changements d'états ou processus caché

Une catégorie morphologique correspond à chacun des mots de la phrase. En parallèle avec les chaînes Y_t, nous pouvons donc envisager des chaînes X_t des catégories morphologiques x_t.

La chaîne totale X_T appelé *chaîne cachée* égale à *pper verb pper infi* va correspondre à l'ensemble des 4 catégories associées aux mots :

x_1 : *pper* (pronom personnel)
x_2 : *verb* (verbe)
x_3 : *pper*
x_4 : *infi* (infinitif)

L'élément générique est x_t avec t compris entre 1 et T=4

On peut considérer des sous-chaînes de la chaîne totale :

X_1 = *pper*
X_2 = *pper verb*
X_3 = *pper verb pper*
X_4 = *pper verb pper infi*

En réalité, le troisième mot présente une ambiguïté : il peut être aussi déterminant *det.*

On peut donc trouver deux propositions :

y_1 : *il*	→	x_1 : *pper*
y_2 : *faudrait*	→	x_2 : *verb*
y_3 : *la*	→	x_3 : *pper*
y_4 : *remonter*	→	x_4 : *infi*

et

y_1 : *il*	→	x_1 : *pper*
y_2 : *faudrait*	→	x_2 : *verb*
y_3 : *la*	→	x_3 : *det*
y_4 : *remonter*	→	x_4 : *infi*

Bien entendu, le but à atteindre est de résoudre l'ambiguïté : dans cette phrase, le mot *la* est-il un pronom personnel *pper* ou un déterminant *det* ?.
Nous devons aussi être capables de faire face à des mots inconnus (oubliés ou non reconnus par les lexiques) ce qui correspond par exemple à :

y_1 : *il*	→	x_1 : *pper*
y_2 : *faudrait*	→	x_2 : *verb*
y_3 : *lla*	→	x_3 : *?*
y_4 : *remonter*	→	x_4 : *infi*

5.2.1.3 Ordre des modèles

Les modèles de Markov appliqués à la désambiguïsation lexicale reposent sur l'hypothèse que dans une suite de catégories morphologiques, une catégorie dépend de celles qui l'entourent.

Par exemple :

> La catégorie du troisième mot *la* dépend des catégories des mots l'entourant : *pper verb* et *infi*.

Ordre des modèles :

- Si la catégorie x_i ne dépend que de x_{i-1}, on dit que le modèle est d'*ordre un* (mémoire d'ordre 1) ou que le modèle est *bi-grammes*.
- Si la catégorie x_i ne dépend que de x_{i-1} et x_{i-2}, on dit que le modèle est d'*ordre deux* (mémoire d'ordre 2) ou que le modèle est *tri-grammes*.
- D'une façon générale, si la catégorie x_i dépend des n catégories précédentes on dit que le modèle est d'*ordre n* (mémoire d'ordre n) ou bien que le modèle est *(n+1)-grammes*.

On appellera *transition* chacun des enchaînements possibles.

Exemples de transitions :

Transitions d'ordre 1 :

pper verb
verb pper
verb det
pper infi
det infi

Transitions d'ordre 2 :

pper verb pper
pper verb det
verb pper infi
verb det infi

5.2.1.4 Matrices de transitions

Nous verrons que des statistiques vont permettre d'attribuer à toute transition une probabilité. L'ensemble de ces probabilités va définir une matrice *appelée matrice de transitions*. Dans CELINE, les matrices de transitions sont valides pour un utilisateur donné.

Transitions d'ordre un à partir de *ver*b :			Transitions d'ordre deux à partir de *det subc* :		
- vers	*adji*	Probabilité : 0,0143	- vers	*adjq*	Probabilité : 0,1778
- vers	*adjq*	Probabilité : 0,0143	- vers	*adv*	Probabilité : 0,0222
- vers	*adv*	Probabilité : 0,0714	- vers	*apdi*	Probabilité : 0,1333
- vers	*ce*	Probabilité : 0,0143	- vers	*coco*	Probabilité : 0,0667
- vers	*det*	Probabilité : 0,2143	- vers	*conp*	Probabilité : 0,0222
- vers	*infi*	Probabilité : 0,2143	- vers	*infi*	Probabilité : 0,0222
- vers	*intj*	Probabilité : 0,0286	- vers	*loca*	Probabilité : 0,0222
- vers	*loca*	Probabilité : 0,0429	- vers	*ne*	Probabilité : 0,0222
- vers	*pas*	Probabilité : 0,0857	- vers	*pper*	Probabilité : 0,1111
- vers	*pnp*	Probabilité : 0,0144	- vers	*prep*	Probabilité : 0,1333

Figure 5.1 : Exemples d'extraits de matrices de transitions

5.2.1.5 Représentation des modèles

On peut représenter les transitions sur un graphe (Figure 5.2) que nous commentons :

- Les ambiguïtés se traduisent par un treillis ; les arcs sont décorés par les probabilités des transitions correspondantes. Les nœuds représentent les états du système depuis un état initial E.I. jusqu'à un état final E.F.
- La première transition reçoit une probabilité 1 car il s'agit d'une pseudo-transition entre l'état initial et un mot non ambigu. Si le premier mot était ambigu, la probabilité affectée pourrait être déduite des probabilités du vecteur de début de phrase (§ 5.4.4).
- Signalons que l'analyseur morphologique utilisé PILAF indique pour le verbe *remonter* uniquement la catégorie *infi et* notons que les catégories lexicales les plus pertinentes pour le système n'étant pas forcément délivrées par les agents lexicaux (ici *infi*) il y aurait lieu éventuellement de filtrer et corriger les émissions (§ 4.2.1.1 portant sur les tables de correspondances entre agents).

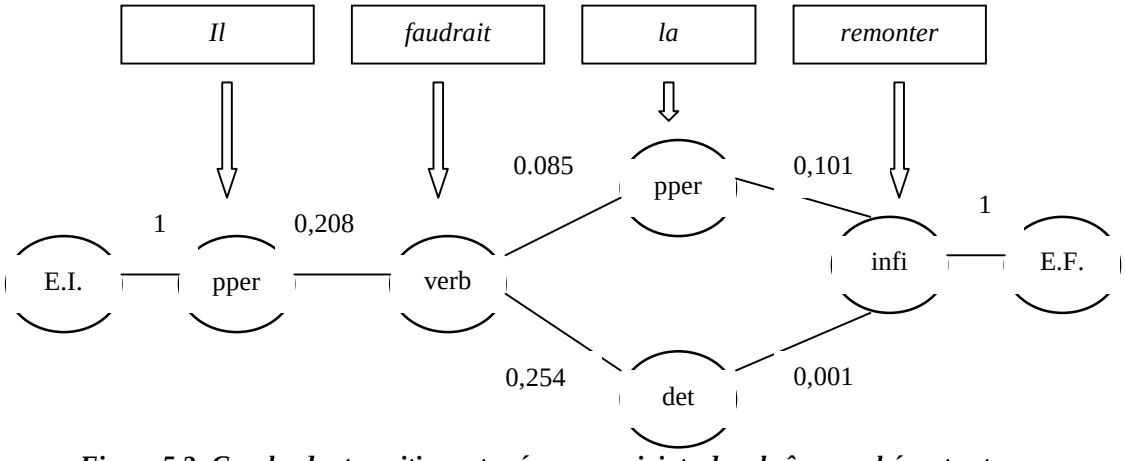


Figure 5.2 :Graphe des transitions et présence conjointe des chaînes cachées et externes

5.2.1.6 Type de modèles

On peut utiliser les modèles de Markov de deux façons différentes :

- La première n'envisage qu'un seul type de chaîne (par exemple la chaîne des catégories morphologiques) et ne prend en compte que les probabilités des transitions de type $x_{t-1}\, x_t$ pour l'ordre 1 ou $x_{t-2}\, x_{t-1}\, x_t$ pour l'ordre 2.
- La deuxième (Modèles de Markov Cachés) va tenir compte de la présence simultanée des deux chaînes (chaînes externes et cachées) ce qui va introduire un terme supplémentaire dit de *probabilité conjointe*.

Dans le prochain paragraphe, nous allons reprendre une formulation très générale des modèles de Markov cachés exposée dans (Pérennou 91).

5.2.2 MODÈLES DE MARKOV CACHÉS

L'exemple du paragraphe d'introduction illustre la signification des notations.

5.2.2.1 Formalisme

Un modèle de Markov caché se caractérise par un système à états comportant deux processus :

- un processus de changement d'états appelé processus caché.
- un processus d'émission appelé processus externe.

Les réalisations du premier processus sont des chaînes cachées :

$X = x_0\, x_1 ... x_t ... x_T\, x_{T+1}$

avec :

x_0 : état initial avant le début des émissions

x_{T+1} : état final

x_t tel que $1 \le t \le N$

Les réalisations du second processus sont des chaînes externes ou observations :

$Y = y_1 ... y_t ... y_T$

avec :

y_t tel que y_t est élément d'un espace d'observation Ω.

Concrètement, les chaînes cachées peuvent être, par exemple les catégories morphologiques, alors que les observations peuvent être les mots de la phrase.

Soit p la loi de probabilité régissant conjointement les deux processus. On peut écrire :

$$p(X_t) = \prod_{s=1}^{t} p(x_s / X_{s-1})$$

$$p(Y_t / X_t) = \prod_{s=1}^{t} p(y_s / X_t, Y_{s-1})$$

$$p(X_t, Y_t) = p(Y_t / X_t)\, p(X_t) = \prod_{s=1}^{t} p(y_s / X_t, Y_{s-1})\, p(x_s / X_{s-1})$$

Un modèle de Markov caché du premier ordre se définit par des conditions sur les lois de probabilités des chaînes cachées et des chaînes externes.

5.2.2.2 Condition sur les chaînes cachées.

La loi de probabilité (marginale) des chaînes cachées vérifie :

$$p(x_t / X_{t-1}) = p(x_t / x_{t-1}) = a(x_{t-1}, x_t)$$

Les transitions sont donc régies par la matrice :

$a = [a(i,j)]$, $0 \le i \le N$ et $1 \le j \le N$

avec la convention : $a(x_0 , x_1) = a(0 , x_1)$.

Remarquons que poser la condition ci-dessus équivaut à dire que les chaînes cachées sont des chaînes de Markov d'ordre un. D'où

$$p(X_u) = \prod_{t=1}^{u} a(x_{t-1}, x_t)$$

Par exemple

a(verb,det) = 0,254
a(det,infi) = 0,001
a(verb,pper) = 0,085
a(pper, infi) = 0,101

La probabilité d'avoir la chaîne cachée *verb pper infi* est donc :
P(verb det infi) = a(verb det) * a(det infi) = 2,54 10^{-4}

5.2.2.3 Condition sur les chaînes externes.

La loi de probabilité d'émission des chaînes externes vérifie :

$$p(y_t/X,Y_{t-1}) = p(y_t/x_{t-1},x_t) = b(x_{t-1},x_t,y_t)$$

Les observations sont donc régies par la matrice :

$$b = [b(i,j,y)] , 0 \leq i \leq N , 1 \leq j \leq N \text{ et } y \in \Omega$$

avec la convention $p(y_1/x_0,x_1) = p(y_1/x_1) = b(0,x_1,y_1)$.

Par exemple :

b(verb, pper, *la*) représente la probabilité que le troisième mot *la* soit un *pper* sachant que le mot précédent est un *verb*.
Nous pouvons remarquer que le fait d'introduire le mot y_3 = *la* limite les candidats x_3 à l'ensemble *det* et *pper*.

La loi d'émission dépend de l'état présent et de l'état précédent ; c'est donc une émission avec mémoire d'ordre 1.
La probabilité d'émission d'une chaîne s'écrit ($1 \leq u \leq T$) :

$$p(Y_u / X_u) = \prod_{t=1}^{u} b(x_{t-1}, x_t , y_t)$$

5.2.2.4 Probabilité conjointe et probabilité totale.

La probabilité conjointe d'une chaîne cachée et d'une chaîne externe s'écrit ($1 \leq u \leq T$) :

$$p(X_u , Y_u) = \prod_{t=1}^{u} a(x_{t-1}, x_t) b(x_{t-1}, x_t , y_t)$$

La probabilité de l'émission s'obtient alors en utilisant la loi de probabilité totale :

$$p(Y_u) = \sum_{X_u} p(X_u) p(Y_u / X_u)$$

ou encore :

$$p(Y_u) = \sum_{X_u} \prod_{t=1}^{u} a(x_{t-1}, x_t) b(x_{t-1}, x_t , y_t)$$

5.2.2.5 Mise en œuvre des modèles

Cette mise en œuvre comporte trois phases :

- Prétraitement statistique (Echantillonnage)
- Calcul des modèles (Apprentissage)

- Résolution automatique (Filtrage). Un filtre statistique d'ordre r correspond à la mise en œuvre d'un modèle Markovien d'ordre r destiné à la résolution automatique des solutions multiples.

Nous allons évoquer ces trois phases au prochain paragraphe en insistant plus spécialement sur les algorithmes de filtrage.

5.3 FILTRES STATISTIQUES MARKOVIENS

5.3.1.1 Première phase : Prétraitement statistique

Elle comprend 2 étapes :

- La préparation du corpus avec mise en évidence des ambiguïtés associées aux formes du corpus par l'analyseur.
- La préparation de l'échantillon initial w_0 réalisé manuellement par intervention d'un expert. Pour chaque ambiguïté, l'expert va choisir une solution unique.
  Par exemple :
  > Si le corpus d'apprentissage contient notre phrase-exemple *il faudrait la remonter* l'expert va choisir la catégorie morphologique *pper* pour le troisième mot.

Le lexique joue un rôle important dans l'apprentissage. Pour chaque mot il doit contenir les classes lexicales, états du modèle, qui peuvent l'émettre (par exemple : <la : det, pper>).

5.3.1.2 Deuxième phase : calcul des modèles

L 'apprentissage peut se faire selon différentes approches :

5.3.1.2.1 Approche statistique

Grâce à l'échantillon w_0 "non ambigu" du corpus, le filtre va acquérir par comptage certaines connaissances sur les différents enchaînements des catégories morphologiques.
Les comptages se traduisent par la matrice de fréquence de transition $N_{ij}(w)$.

A partir de cette matrice on peut ensuite :
- estimer les paramètres de la chaîne de Markov,
- conclure sur la représentativité de l'échantillon.

Ces connaissances sont toutefois fonction de l'échantillon w_0 proposé.

Les utilisateurs des modèles de Markov disposant de corpus de tailles suffisantes créent un modèle de départ approximatif soit par un sous-corpus étiqueté soit en se donnant des probabilités subjectives. Ils alignent ensuite sur un corpus plus grand non étiqueté et obtiennent alors un deuxième modèle au prix de quelques contrôles manuels. Au bout de quelques dizaines d'itérations au plus le modèle est stabilisé et le corpus est étiqueté.

Dans notre cas, le problème est différent : le corpus se construit d'une manière incrémentale à partir des travaux du rédacteur et les tailles insuffisantes des corpus utilisés entraînent une mauvaise convergence des probabilités (§ 5.4.5). De plus les interventions manuelles pour « arranger les modèles » sont par nature impossible.

Pour un rédacteur donné, ces connaissances pourront toutefois être évolutives en accumulant les données acquises a posteriori pour d'autres échantillons.

Cette méthode peut constituer la première étape de méthodes d'apprentissage telles que celles présentées ci-après.

5.3.1.2.2 Approche fondée sur la théorie de l'information

Cette approche (exposée de même que la suivante dans (CALLIOPE 89) est basée sur la maximisation d'une fonction appelée fonction auxiliaire définie par

$$Q(\theta, \theta') = \sum_{\sigma} p((O, \sigma) \mid \theta) \log(p(O, \sigma) \mid \theta')$$

avec

O	une suite d'observations (corpus d'apprentissage),
σ	la suite de transitions $E_{t-i}E_t$,
θ	ancienne valeur du paramètre,
θ'	nouvelle valeur du paramètre.

Soit T la transformation qui à θ associe le θ' qui réalise ce maximum, l'itération de cette transformation T converge vers un maximum local de $p(O \mid \theta)$.

5.3.1.2.3 Approche dite par Optimisation

La méthode consiste à maximiser la fonction $\text{Max}_{\theta}\, p(O \mid \theta)$ avec des contraintes sur θ s'il y en a.

5.3.1.3 Troisième phase : résolution automatique

Pour chaque ambiguïté du corpus, on cherche à obtenir par exemple:

- les solutions les plus "probables"
- les autres solutions classées par ordre décroissant de probabilité
- les intervalles de confiance des solutions les plus probables, ce qui permet de définir une stratégie à leur égard.

Nous examinerons les spécifications d'un filtre destiné à être intégré dans le système CELINE en distinguant deux finalités différentes :

- aider le correcteur orthographique en proposant des catégories morphologiques ou en validant des catégories proposées par d'autres agents. Le filtre effectue alors un travail local sur quelques mots.
- aider l'analyseur syntaxique en diminuant le degré d'ambiguïté du corpus. Le filtre travaille alors sur la totalité du corpus. Cette finalité nous ramène à une problématique du type RAP.

5.3.1.4 Algorithme de BAUM-WELCH

5.3.1.4.1 Principe

Cet algorithme traditionnel (Calliope 89) (Pérennou 91) est à placer un peu à part. En effet, il peut servir dans la deuxième phase (calcul des modèles) ou dans la troisième phase (résolution automatique).
Cet algorithme est encore appelé algorithme E-M (Expectation - Modification) : en observant un échantillon O, on calcule des valeurs moyennes (expectation) à partir de l'estimation θ.

D'une manière itérative (modification), on calcule de nouveaux paramètres θ' au moyen de la formule de ré-estimation.
En un nombre fini d'itérations, il y a convergence assez rapide des paramètres.

5.3.1.4.2 Algorithme Forward

Posons $\alpha(t,x) = p(\xi_t = x, Y_i)$ la probabilité pour que :

l'émission jusqu'à t soit Y_t

l'état atteint au même moment soit $\xi_t = x$

Appliquons la formule de la probabilité d'émission

$$\alpha(t,x)= \sum_{Xt} \prod_{s=1}^{t} a(x_{s-1}, x_s)\, b(x_{s-1}, x_s, y_s)$$

On en déduit les formules de récurrence :

$$\alpha(1, x) = p(\xi_1 = x, y_1) = a(0, x)\, b(0, x, y_1)$$

$$\alpha(t, x) = \sum_{X'} \alpha(t-1, x')\, a(x', x)\, b(x', x, y_t)$$

$$\alpha(T+1, F) = p(\xi_T = F, Y) = \sum_{X'} \alpha(T,x')\, a(x',x)$$

$$p(Y) = \alpha(T+1, F)$$

La formule précédente permet de calculer p(Y) quand on sait que l'émission s'est arrêtée après Y.

5.3.1.4.3 Algorithme Backward

Posons $\beta(t,x) = p(Y^{t+1} / \xi_t = x)$

avec la convention $Y^{t+1}=Y_{t+1}Y_{t+2}...Y_{T+1}$

$$\beta(T, x) = p(Y^{T+1} = \phi / \xi_T = x) = a(x, F) \text{ avec } \phi \text{ chaîne vide}$$

$$\beta(t, x) = \sum_{X'} a(x, x')\, b(x, x', y_t)\, \beta(t+1, x')$$

$$\beta(0, 1) = \sum_{X'} a(I, x')\, b(I, x', y_1)\, \beta(1, x')$$

$$p(Y) = \beta(0, I) = p(Y/ \xi_0 = I)$$

La formule précédente permet de calculer p(Y) si l'on sait que le système est à l'état I avant l'émission de Y.

5.3.2 CONSTRUCTION DE LA MATRICE DE TRANSITION

On applique la propriété $\sum_j P_{ij} = 1$ ce qui permet de construire la matrice de transition

P

$$P_{ij} = \frac{N_{i,j}(w)}{N_i(w)}$$

où N_i est le nombre total de transitions à partir de la catégorie CM_i

$$N_i(w) = \sum_{j=1}^{m} N_{i,j}(w)$$

L'algorithme précédent initialise la matrice de transition de sens direct. On peut, en transposant cette matrice, étudier les transitions de sens inverse entre catégories morphologiques (droite à gauche).

5.3.3 ALGORITHMES DE FILTRAGE

5.3.3.1 Définitions

Aux formes textuelles $f_1,f_2,....f_t$ du corpus analysé CR, l'analyseur associe respectivement les catégories morphologiques $C_1,C_2,....C_t$ avec C_i tel que

$C_i=\{CM_1, CM_2,...CM_k\}$ est un ensemble non vide.

Nous appellerons catégorisation CG la suite $C_1, C_2,....C_t$

La catégorisation est dite ambiguë si l'analyse est ambiguë.

Le nombre des solutions présentes dans une catégorisation CG est donné par

$card(C_1) * card(C_2) *... card(C_t)$

Une zone non ambiguë est une sous-liste $C_i.....C_{i+j}$ de la catégorisation CG telle que :

$\forall k \in [i...i+j] , card(C_k) = 1$

Une zone non ambiguë est appelée un nœud.

Une zone ambiguë est une sous-liste $C_i... C_{i+j}$ de la catégorisation CG telle que :

$\forall k \in [i...i+j] , card(C_k) > 1$

Une zone ambiguë est appelée ventre.

5.3.3.2 Application de l'algorithme de Baum-Welch

Nous avons déjà dit (§ 5.3.1.4) que l''algorithme de Baum Welch permet de rechercher le modèle minimisant la probabilité d'erreur.

$W = argmax[p_v(Y) / v \in V]$

5.3.3.3 Algorithme de Viterbi

5.3.3.3.1 Algorithme

Cet algorithme (Pérennou 91) permet la recherche de la chaîne la plus vraisemblable. Cette chaîne s'obtient par :

$X^* = argmax_X\{p(X) P(Y / X)\}$

la recherche se fait sur le calcul des probabilités maximales $\gamma(t, x)$ des chaînes X_t dont le dernier état est $x_t = x$ correspondant à Y_t soit :

$\gamma(t,x) = max[p(X_{t-1}x, Y_t) / X_{t-1}]$

Formules de récurrence :

$\gamma(1, x) = p(x, y_1) = a(0, x) b(0, x, y_1)$

$\gamma(t, x) = max[\gamma(t-1, x') a(x', x) b(x', x, y_1) / x']$

$P(X^*, Y) = \gamma(T+1, F) = max [\gamma(T, x) a(x, F) / x \in E]$

La solution X* s'obtient en sauvegardant durant la récurrence le pointeur arrière lien(t,x) donné par :

$lien(t, x) = argmax_{x'}[\gamma(t-1, x') a(x', x) b(x', x, y_t)]$

La reconstruction de X* s'effectue à partir de $x^* = argmax_x\{\gamma(T, x)\}$ en suivant le cheminement défini par lien(t, x).

5.3.3.3.2 Conditions d'applicabilité de cet algorithme

Cet algorithme suppose l'utilisation des modèles de Markov d'ordre un puisque la formulation (termes a(x', x) et b(x', x, y_t)) repose sur cette hypothèse. Il ne peut donc s'appliquer à l'ordre deux. (Kriouille 90) a tenté une adaptation de cet algorithme à l'ordre deux (distance de Viterbi à optimalité locale).
Les matrices de transitions doivent être stables ce qui demande une parfaite connaissance de l'utilisateur (échantillonnage sur des corpus de grandes tailles).

5.3.3.3.3 Explication de l'algorithme de Viterbi sur un exemple

Reprenons l'exemple :

Il faudrait la remonter
dont la désambiguïsation correspond à
pper verb pper infi
avec une ambiguïté sur *la* soit *det* soit *pper*

Nous obtenons les étapes suivantes :

γ(1,x) = ?
Le premier mot *il* a pour catégorie *pper* donc x = pper et
γ(1,pper)=a(0,pper)*b(0,pper,il)

γ(2,x) = ?
Pour x seul *verb* est possible donc x=*verb* et
γ(2,verb) = γ(1, pper) a(pper, verb) b(pper, verb, faudrait)

γ(3,x) = ?
Pour x deux solutions sont possibles x = *det* ou x = *pper*
Il faut donc faire intervenir deux dérivations possibles et retenir en final celle donnant la probabilité maximale.

γ(2,verb) a(verb, det) b(verb, det, la)
et de
γ(2,verb) a(verb, pper) b(verb, pper, la)

γ(4, x) = ?
Pour x une seule solution est possible *infi*
Il faut donc prendre le maximum de

γ(4,infi) = γ(3,det) a(det, infi) b(det, infi, remonter)
et de
γ(4,infi) = γ(3,pper) a(pper, infi) b(pper, infi, remonter)

Le calcul numérique et la reconstitution de la chaîne solution conduiront à la bonne solution : *pper verb pper infi*

5.3.3.4 Algorithme de Kallas

Kallas (Kallas 87) utilise les modèles de Markov en ne considérant que les chaînes cachés d'ordre un des catégories morphologiques.
Le filtre étant d'ordre 1, les ventres sont indépendants du point de vue du calcul de probabilités. En effet, la mémoire du modèle ne dépend que de la dernière catégorie rencontrée c'est-à-dire du nœud précédent.
On va donc réaliser un filtrage "ventre par ventre".

Algorithme *filtrage*
début

```
        Positionner_la_première_zone {traitement zone par zone};
        Tant que (zone <> zone_finale) faire
                début
                        | si (zone = nœud) alors
                        |         traiter_nœud(zone)
                        | sinon
                        |         traiter_ventre(zone);
                        | fsi;
                        | zone_suivante;
                fin;
fin.
```

avec

```
Algorithme traiter_nœud(zone)
début
        recopier_la_zone_telle_qu'elle_est(zone);
fin;

Algorithme traiter_ventre(zone)
début
        | dénombrer_les_solutions_possibles;
        | calculer_la_probabilité_de_chaque_solution;
        | classer_les_solutions_par_ordre_décroissant_de_probabilité;
        | calculer_l'intervalle_de_confiance_de_la_solution_la_plus_probable(Ip max);
        | choisir_la_solution_la_plus_probable {sortie};
        | signaler_les_solutions_concurrentes;
        | sauvegarder_les_résultats {sorties};
fin;
```

Cas particulier :

L'algorithme suppose que la première forme est non ambiguë.

Si ce n'est pas le cas, on peut :

- ou bien lever l'ambiguïté manuellement,
- ou bien ajouter comme état initial un nœud fictif CM_0 de probabilité 1 en tête.

$Pr(CM_1) = Pr(CM_0)*Pr(CM_1/CM_0)$

avec $Pr(CM_0)=1$.

Cet algorithme sous-optimal peut venir en remplacement de l'algorithme de Viterbi dans le cas où on cherche à faire, non pas une désambiguïsation complète de corpus mais une résolution locale.

De plus, son adaptation à l'ordre deux est évidente en considérant comme nœud un ensemble de deux mots reconnus non ambigus ou dont les ambiguïtés ont étés levées par un procédé autre (analyse syntaxique par exemple).

5.3.4 CONCLUSION

Nous avons présenté quelques méthodes de filtrage. Nous allons rechercher maintenant nos spécificités afin de justifier les choix retenus pour Céline (§ 5.5). Les agents réalisés sur ces bases nous ont donné satisfaction et pour l'instant nous n'avons pas poussé au-delà notre investissement.

5.4 RESULTATS EXPERIMENTAUX

Ces résultats sont fondamentaux puisqu'ils vont être à l'origine des choix.

5.4.1 ORDRE DES CHAÎNES DE MARKOV ET TYPE DE CORPUS

5.4.1.1 Rappel de résultats concernant l'écrit.

Dans le domaine de l'écrit, les différents expérimentateurs sont en général unanimes pour considérer que l'ordre optimum des chaînes de Markov des catégories morphologiques est de deux (Modèle dit tri-classes). Nous ne reviendrons pas sur ce point que nous considérerons comme acquis.

Par exemple, Kallas (Kallas 87) donne les éléments suivants:

Validation théorique par le test du Khi 2 :

L'hypothèse nulle r=1 contre l'alternative r=2 pour un seuil de signification 0,95 est rejetée.

L'hypothèse nulle r=2 contre l'alternative r=3 pour le même seuil de signification est acceptée.

Résultats expérimentaux :

Kallas défini un taux d'erreur par comparaison entre les catégories déterminées par le filtre et les catégories retenues par un expert :

Taux d'erreur = (nombre de catégories fausses)/nombre de catégories totales déterminées.

Taux d'erreur de 3,6 % avec un filtre d'ordre 1.

Taux d'erreur de 1,7 % avec un filtre d'ordre 2.

Autres éléments :

Le traitement direct (Gauche-Droite) ou le traitement inverse (Droite-Gauche) donnent des résultats identiques. Certaines ambiguïtés résolues avec le filtre d'ordre un ne sont pas résolues avec le filtre d'ordre deux et vice-versa.

Enchaînement des deux filtres :

Taux d'erreur de 1,7 % encore, les erreurs ne semblant pas propres à un filtre donné.

Conclusion :

Kallas conclut à la difficulté de diminuer ce taux d'erreur, l'analyse lui ayant montré que certaines ambiguïtés sont liées à des constructions syntaxiques particulières ayant une prépondérance faible.

5.4.1.2 Ordre optimal des chaînes de Markov pour des textes fortement entachés d'erreurs.

La plupart des phrases présentent des discontinuités dans la structure syntaxique. Elles sont souvent découpables en plusieurs "tronçons" correspondant chacun à une construction syntaxique cohérente. La modélisation avec des chaînes longues devient approximative et engendre des erreurs provenant de la présence de nombreux enchaînements à faible probabilité (transitions entre deux tronçons) avec pour conséquence une non-sélectivité du calcul des intervalles de confiance.

En conséquence, il est logique de prendre pour hypothèse que pour les textes, avec des fautes de syntaxe courantes, l'ordre optimum des chaînes de Markov diminue et devient donc de un. Les résultats expérimentaux confirment cette hypothèse.

Nous pouvons par ailleurs constater que la majorité des systèmes de RAP utilisant les modèles de Markov envisage des modèles d'ordre un (possibilité de transformation canonique) et utilisent l'algorithme de Viterbi.

5.4.2 LES LIMITES DES MODÈLES DE MARKOV

Dans le cas d'un filtrage interactif, on retient un certain nombre de solutions concurrentes en appliquant, par exemple, la technique des intervalles de confiance. Dans le cas d'un filtrage automatique, on retient ou bien la solution présentant le maximum de probabilités de vraisemblance ou bien la solution présentant le minimum de probabilités d'erreurs.

Dans les conditions de l'expérimentation, il est évident que le filtre automatique basé sur les modèles de Markov ne peut être efficace à 100%. Par exemple, en l'appliquant sur son propre échantillon, il rejette un certain nombre de solutions à faible probabilité pourtant bien présentes.

Prenons un exemple qui montre les limites de la méthode des chaînes de Markov et la nécessité de procédés complémentaires.

Pour la phrase
" il faudrait la remonter "
dont la transduction morphologique correcte est
" *pper verb pper infi* "
l'analyseur morphologique proposera deux solutions :
" *pper verb pper infi* " et " *pper verb det infi* ".
Si on considère des modèles de Markov d'ordre un, la technique utilisée peut produire des résultats différents :

- Traitement par la technique des intervalles de confiance :

Si après la catégorie *verb* (verbe), l'analyseur morphologique propose les catégories *det* (déterminant) et *pper* (pronom personnel), le filtre statistique d'ordre un va sélectionner *det* quelle que soit la catégorie suivante. Le filtre engendrera l'erreur de sélection :
" *pper verb det infi* ".

Nous obtenons en effet les résultats suivants :

- Probabilité de la transition (*verb, det*) : $P_1 = 0{,}2542$
- Intervalle de confiance de la transition (*verb, det*) :

Cet intervalle se détermine par la formule

avec

$P = 0{,}2542$ probabilité de la transition (*verb, det*)
$N = 59$ fréquence absolue des transitions à partir de *verb*
$u_0 = 1{,}96$ pour un seuil de confiance de 95%

On obtient 0,0566 comme valeur de l'intervalle de confiance.

- Probabilité de la transition (*verb, pper*) : $p_2 = 0{,}0847$

Nous nous apercevons que p_2 est nettement en dehors de l'intervalle $\{P_1\text{-}I/2,\ P_1\text{+}I/2\}$ soit {0,2259 ; 0,2825}. La transition (*verb, pper*) ne peut donc pas être considérée comme une solution concurrente de la transition (*verb, det*). Cette solution serait donc rejetée, même par un filtre interactif.

L'intérêt de cette technique est qu'elle ne fait pas intervenir la catégorie du mot suivant *infi* car, dans notre problématique de la correction d'erreurs, le mot suivant pourrait être totalement indéchiffrable.

- Traitement par l'algorithme de Viterbi

Nous avons vu au § 5.3.3.3 que pour le même exemple, cette méthode conduisait à retenir la solution correcte.

Bien qu'utilisant dans les deux cas les modèles d'ordre un, nous pouvons attribuer cette différence au fait que les probabilité des transitions *det infi* et *pper infi* vont être prises en compte.

La force et l'optimalité de l'algorithme de Viterbi résulte du principe de décision retardée : on ne décide rien avant d'avoir tout fait intervenir : c'est ainsi que l'aspect local des bigrammes n'est pas incompatible avec la structure locale de la phrase.

5.4.3 SOLUTIONS CONCURENTES DU FILTRE DE MARKOV D'ORDRE 1

Le filtrage inverse, le retour arrière ou le filtre d'ordre deux, accompagnés ou pas d'un calcul de probabilité du chemin, permettent de résoudre certaines ambiguïtés incorrectement résolues par le filtre d'ordre un.
Prenons par exemple, le calcul de la probabilité du chemin dans le cas précédent. On obtient :

Probabilité de la transition (*verb, det*) : 0,254
Probabilité de la transition (*det, infi*) : 0,001
Probabilité du chemin : 0,254*0,001 = 0,003

Probabilité de la transition (*verb, pper*) : 0,085
Probabilité de la transition (*pper, infi*): 0,101
Probabilité du chemin : 0,085*0,101 = 0,085

Cette fois, le calcul probabiliste donne bien le résultat exact.

Malheureusement cette solution entraîne parfois des erreurs sur des enchaînements que le filtre d'ordre un résout correctement (principalement à cause de la fragmentation des phrases).

Une autre solution consiste à tenir compte du contexte. Dans le cas de l'exemple précédent, l'enchaînement *verb det infi* (ou *verb* et *infi* forment le contexte de *det*) correspond à un enchaînement inacceptable par un linguiste. Un problème est que, dans le filtrat morphologique, le contexte n'est pas encore constitué. Il faut donc pratiquer un filtrage à deux étages et appliquer sur le filtrat morphologique d'abord le filtre de Markov et ensuite un filtre grammatical contextuel.

5.4.4 LE PROBLÈME DES DÉBUTS DE PHRASE

La transduction morphologique du premier mot de chaque phrase pose un problème particulier. En effet si cette transduction est ambiguë, l'initialisation de la chaîne de Markov ne peut pas se faire.

Pour lever les ambiguïtés de début de phrase, on peut envisager différentes solutions :

1) Levée d'ambiguïté interactive (difficilement admissible pour un filtre automatique !).

2) Application du vecteur de probabilité de début de phrases pour un rédacteur particulier.
L'idée de ce vecteur est basée sur trois remarques:

a) si la construction des phrases est parfois globalement décousue, la construction de chaque tronçon est la plupart du temps syntaxiquement correcte (y compris le tronçon en tête de phrase).

b) les constructions syntaxiques changent d'un individu à l'autre mais demeurent souvent constantes pour un même individu. L'extrait de texte de l'annexe B montre, par exemple, un nombre élevé de phrases débutant par le pronom personnel "je" suivi d'un verbe. En fait, sur le total du texte correspondant, on s'aperçoit que sur les 61 catégories morphologiques utilisées, seules 11 figurent en début de texte. De plus, elles ne sont pas en général "trop" concurrentes les unes des autres.
Si sur N phrases, le nombre d'occurrences en tête de phrase de la catégorie morphologique "cat" est de n, la probabilité associée est définie par P(cat)=n/N.
La levée d'ambiguïté consiste tout simplement à retenir parmi les solutions concurrentes, la plus probable. Très efficace pour la "pensée orale", jusque-là aucune erreur résiduelle ne lui étant imputable, cette technique pourrait toutefois être mise en échec par un locuteur utilisant en début de phrase, une bonne partie de l'ensemble des catégories morphologiques.

c) Ce vecteur traduit conformément au théorème des probabilités composées le terme $Pr(E_0)$.

3) Parcours de la phrase jusqu'à trouver un mot non ambigu (s'il existe !) et à partir de ce mot résolution de la tête de phrase par filtrage de sens inverse et résolution de la queue de phrase par filtrage en sens direct. Ce procédé efficace sur des phrases bien construites devient plus incertain dans le domaine de la "pensée-orale" ou des textes fortement entachés d'erreurs syntaxiques car le premier mot non ambigu trouvé dans la phrase peut très bien faire partie d'un tronçon de phrase discontinue avec le début de phrase (par exemple si un mot d'une phrase ou un séparateur entre deux phrases ont été oubliés).

4) Calcul du poids des chemins différents possibles et conservation du plus probable.

Par ailleurs, l'utilisation du vecteur de début de phrase donne de bons résultats pour certains cas particuliers tels que:

- phrase composée d'un seul mot.
- phrase pour laquelle tous les mots ont une transduction ambiguë.

5.4.5 TAILLE DU CORPUS ÉCHANTILLON ET VALIDITÉ

5.4.5.1 Critère de validité

Pour une estimation correcte, le nombre de données (taille du corpus) va dépendre du nombre de paramètres.

Le nombre N de catégories dans l'échantillon n'est pas déterminé a priori.

- Lorsque N est petit, le modèle obtenu engendre beaucoup plus de chaînes qu'il n'en existe dans l'ensemble d'apprentissage (phénomène dit du *surapprentissage*).
- Lorsque N est très grand, le modèle de Markov correspondant rend compte des exemples d'apprentissage et d'eux seuls.

On recherche d'une manière empirique une valeur intermédiaire produisant une optimisation de la levée des ambiguïtés.

Une règle empirique admise consiste à dire que le nombre de l-gramme doit être supérieur à $K * m^l$,

m étant le nombre de catégories morphologiques,

l étant la longueur de la chaîne de Markov soit 2 pour l'ordre un et 3 pour l'ordre deux et les modèles tri-classes.

K une constante multiplicative. Dans le paragraphe suivant nous prendrons K=1 pour obtenir un ordre de grandeur (sans plus).

Selon la taille du corpus, le nombre de l-grammes observé va varier. Les l-grammes non observés se verront affectés une fréquence de 1 ce qui permet de leur affecter une probabilité petite mais non nulle.

5.4.5.2 Étude de la validité du corpus en fonction du nombre de catégories morphologiques :

L'annexe B présente un exemple de textes utilisés pour tester les capacités de Markov à pouvoir proposer une catégorie morphologique pour un mot inconnu en fonction du rédacteur. Le tableau de la figure 5.2 présente la taille des textes et du corpus ; ce tableau a été réalisé avec un paramétrage et une désambiguïsation manuelle comportant 56 catégories morphologiques.

Le corpus - permet donc d'étudier l'ordre un : $56^2 = 3136$,
- ne permet pas d'étudier l'ordre deux : $56^3=175616$.

Pour l'ordre 2, il faudrait créer suffisamment de super classes de manière à tomber au plus à $(26554)^{1/3}=29,8$ soit une trentaine de catégories morphologiques.

Nom du fichier	Nombre de mots
BNDIS	1295
ERCIP	1412
ERCIS	1422
HLNIS	3147
ISAIP	3166
JCLIP	1934
JEN2IS	2100
JENLIS	1562
JNYIP	2790
JNYIS	1778
MLUIP	1706
MRTIP	1017
MRTIS	2547
MURIP	678
TOTAL : 26554 mots	

Figure 5.2 : Taille des textes échantillons

Pour envisager des chaînes plus longues, le nombre de catégories morphologiques devrait encore baisser :

$(26554)^{1/4}=12,7$ soit 12 catégories morphologiques.

$(26554)^{1/5}=7,6$ soit 7 catégories morphologiques.

Aucun texte pris séparément n'est de longueur suffisante pour l'ordre un, et a fortiori pour l'ordre deux.

Remarquons que, si ce fait intervient directement dans le choix des méthodes pouvant être utilisées, il était prévisible par avance. L'idée de mener une expérimentation permettant de trancher sur la possibilité de se dispenser d'un "coûteux" échantillonnage en découle.

5.4.5.3 Cumul, convergence et solution algorithmique retenue

Afin de tester les algorithmes vérifiant la convergence, le texte le plus long ISAIP a été découpé en sous textes d'environ 400 mots chacun. La stabilisation des valeurs des matrices que nous appellerons *convergence* n'a pas été observée.

Le cumul des transitions sur des locuteurs différents ne donne pas de résultats fiables non plus.

Une amélioration consiste à regrouper les locuteurs par niveau d'expertise et "façon" de s'exprimer. Dans ce cas on observe un début de convergence.

Nous lions cette absence de convergence à la taille trop petite de notre corpus[4] (paragraphe ci-dessus). Nous n'avons pas cherché à tester les modules avec des corpus plus importants car cela ne nous semblait pas compatible avec le cadre général de CELINE destiné à s'intéresser à un rédacteur particulier. Pour faire une comparaison, le présent chapitre de 30 pages comprend environ 10 000 mots (statistique Word). Notre corpus de 30 000 mots représente donc une centaine de pages désambiguïsées.

En conséquence nous admettons définitivement le choix supplémentaire pour CELINE :

1. *En ce qui concerne l'échantillonnage :*
   - *Utilisation d'un algorithme de comptage statistique simplifié*
   - *Rejet pour l'apprentissage d'algorithme plus séduisant du type Baum-Welch*
2. *En ce qui concerne le filtrage :*
   - *Utilisation des modèles de Markov d'ordre un pour un rédacteur peu performant ou pour des textes issues de l'oral. Si possible (connaissance du rédacteur) l'utilisation de l'algorithme de Viterbi.*
   - *Utilisation des modèles de Markov d'ordre deux pour un rédacteur performant.*

5.4.5.4 Exemple de résultats

5.4.5.4.1 Résultats sur un corpus de 678 mots

Nombre de catégories morphologiques : 61
Nombre de substitutions effectuées : 4
Nombre théorique de types de transitions d'ordre 1 : 3249
Nombre théorique de types de transitions d'ordre 2 : 185193

Nombre de types de transitions d'ordre 1 observées : 163

Nombre théorique de types de transitions d'ordre 2 compte tenu du nombre réel de transitions d'ordre 1 : 9291

Nombre de types de transitions d'ordre deux observées : 303

5.5 UTILISATION DES MODÈLES DANS CELINE

5.5.1 FINALITÉS

Les finalités peuvent être variables :

[4] Nous devons mener en 99 une expérimentation analogue avec le corpus GRACE (deux textes d'environ 25 000 et 110 000 mots).

- diminuer le degré d'ambiguïté de manière à simplifier l'analyse syntaxique,
- résoudre localement mais complètement sur quelques mots par exemple pour pouvoir faire face à une forme textuelle inconnue à proximité (voir paragraphe suivant)
- Répondre à une demande d'aide d'un des autres agents de CELINE

Pour chaque ambiguïté du corpus, on obtient :

- la solution la plus "probable" (ou les solutions si probabilités ex-æquo),
- les autres solutions classées par ordre décroissant de probabilité,
- le(s) intervalle(s) de confiance de la (ou des) solution(s) la (ou les) plus probable(s), ce qui permet de définir une stratégie à son (leur) égard. On appelle solutions concurrentes des solutions telles que leurs probabilités sont comprises dans ces intervalles.

Dans l'hypothèse d'un filtrage automatique complet, nous choisirons de retenir la ou les solutions les plus probables. Nous compléterons l'élimination des solutions concurrentes par coopération en nous servant des autres niveaux d'analyse (voir le chapitre sur CELINE).

5.5.2 RÈGLES GÉNÉRALES APPLICABLES

Nous allons tout d'abord récapituler et préciser nos conclusions des paragraphes précédents. Nous détaillerons ensuite quelques-unes des stratégies possibles.

1 Règles stratégiques générales :
- Utiliser les modèles de Markov pour la proposition de catégories morphologiques pour une forme textuelle inconnue.
- Utiliser les modèles de Markov pour départager des catégories morphologiques déjà proposées par d'autres agents.
- Rester prudent en ce qui concerne le rejet de solutions à partir des seuls aspects quantitatifs des modèles.
- Utiliser la technique des intervalles de confiance pour désambiguïser au moins partiellement.

2 Règles heuristiques d'emplois des modèles de Markov selon la compétence de l'utilisateur telle que :
- Utiliser le modèle bi-classe pour un rédacteur à faible compétence ou le modèle tri-classe pour un rédacteur à compétence élevé.
- Choix de seuils différents selon la compétence du rédacteur et le modèle (ordre un ou ordre deux)
- Utiliser le vecteur de probabilité de début de phrase.

3 Règles stratégiques en cas de difficultés c'est-à-dire quand la démarche normale ne permet pas une désambiguïsation suffisante :
- Assistance du mode parcours de sens direct par le mode parcours de sens inverse.

- Assistance du modèle d'ordre un par le modèle d'ordre deux et vice versa.
- Parcours en sens direct ou inverse et comparaison des deux ensembles de solutions S_1 et S_2 sans seuil de sélection ni application de la technique des intervalles de confiance mais avec éventuellement construction d'un nouvel ensemble S_3 des solutions communes.
- Le *mode secours* **:** on abaisse tous les seuils à zéro et on accepte tout ce que produit le filtre, les autres agents se débrouilleront !

4 Règle de fin d'apprentissage pour un rédacteur

- La mémorisation des fréquence $N_{i,j}$ est implantée sur des entiers sur deux octets soit une valeur maximale de $2^{16} - 1 = 65535$.
- L'apprentissage se termine dès que l'une quelconque des fréquences atteint la valeur 65535.

5.5.3 STRATÉGIE VARIABLE DANS LES VENTRES D'AMBIGUÏTÉS

5.5.3.1 Cas d'une phrase comportant une forme textuelle inconnue

Dans ce cas, l'objectif est de proposer une catégorie morphologique pour la forme textuelle inconnue. CELINE va procéder à un filtrage en trois étapes :

1. Désambiguïsation si possible du contexte droit et gauche de la forme textuelle inconnue comme indiqué dans le paragraphe précédent.

et sous réserve du succès (au moins sur un côté et au moins du modèle d'ordre 1) de cette première étape :

2. Restriction du nombre de solutions demandées à la modélisation de Markov pour la forme inconnue en utilisant la technique des seuils de probabilité.
    - Le cas idéal correspond au cas d'une solution de probabilité élevée avec toutes les autres solutions en dehors de l'intervalle de confiance. Les modèles de Markov fournissent alors une seule solution et la désambiguïsation est totale
    - Un cas déjà plus délicat correspond à plusieurs solutions acceptables dans l'intervalle de confiance de la solution de probabilité la plus élevée. Le système va prendre en compte chacune des solutions concurrentes. Les modèles ont simplement contribué à la diminution du degré d'ambiguïté dans ce cas. Cet aspect n'est pas négligeable.
    - Cas d'échec : aucune proposition. Dans ce cas, ou bien l'échantillonnage n'a pas été suffisant ou bien les seuils sont trop hauts. On va alors changer de stratégie et passer en *mode secours*.
3. Coopération des autres agents sur les solutions éventuelles résiduelles concurrentes. Cette troisième étape sera exposée dans le chapitre sur CELINE.

Premier cas : pour commencer nous supposerons une seule forme textuelle inconnue ou des formes textuelles inconnues non adjacentes :

mot 1	mot 2	Forme textuelle inconnue	mot 3	mot 4
cat 1	cat 2	X	cat 3	cat 4

Le filtre va alors proposer une ou plusieurs solutions dépendant du modèle choisi et du sens. La technique adoptée dans CELINE est de fixer des seuils de probabilité distincts pour les modèles bi-classes (seuil sb) et tri-classes (seuil st).

Conformément à nos choix de règles du modèle probabiliste on prendra :

- pour un rédacteur à haute compétence un seuil sb élévé et un seuil st moyen,
- pour un rédacteur à faible compétence un seuil sb faible et un seuil st élevé.
- les seuils ne sont pas affectés arbitrairement mais dépendent du rédacteur par l'intermédiaire de la matrice de transition. La référence est s_0 moyenne arithmétique des probabilités de la matrice de transition d'ordre correspondant. Quelques choix testés :

- seuil faible = s_0, seuil moyen = $2*s_0$, seuil élevé =$3*s_0$.
- seuil faible = s_0, seuil moyen = 0.5, seuil élevé = 1 - s_0

Le filtre proposera donc par exemple :

- modèle bi-classe
  - direct
    - cat 2 solution 1 probabilité p1 avec p1>sb
    - cat 2 solution 2 probabilité p2 avec p2>sb
  - inverse
    - cat 3 solution 3 probabilité p3 avec p3>sb
- modèle tri-classe
  - direct
    - cat 1 cat 2 solution 4 probabilité p4 avec p4>st
    - cat 1 cat 2 solution 5 probabilité p5 avec p5>st
  - inverse
    - solution 6 cat 3 cat 4 probabilité p6 avec p6>st

et pour la probabilité la plus élevée : l'intervalle de confiance.

Deuxième cas : supposons des formes textuelles inconnues adjacentes :

mot 1	mot 2	Forme textuelle inconnue G	Forme textuelle inconnue D	mot 3	mot 4
cat 1	cat 2	X G	X D	cat 3	cat 4

Le choix adopté dans CELINE est de les traiter indépendamment en prenant le contexte inverse droit cat 3 cat 4 pour la forme textuelle droite et le contexte direct gauche pour la forme textuelle gauche.

En cas d'échec d'un des côtés, on résout en se servant de la catégorie résolue de l'autre forme (cas séquentiel ci-dessous).

Dernier cas : une ou deux formes textuelles inconnues en début de phrase.

Forme	Forme		

textuelle inconnue G	textuelle inconnu D	mot 1	mot 2
X G	X D	cat 1	cat 2

Dans ce cas CELINE va utiliser deux techniques différentes :

1. Utilisation du vecteur de probabilité de début de phrase pour X G et ensuite détermination comme au cas 1 de X D
2. Résolution séquentielle :
   Recherche de X D
   Puis pour chaque valeur de X D candidate, recherche en inverse de X G

Le rôle du filtre s'arrête à des propositions au-delà des seuils et il ne lui appartient pas de résoudre complètement le problème qui sera laissé à CELINE par coopération avec d'autres agents.

5.5.3.2 Phrases complètement ambiguës

Une phrase complètement ambiguë ne contient aucun nœud.
Par exemple PILAF va donner de la phrase « *voilà la nuit noire tombe aussi* » la transduction morphologique :

{	{	{	{	{	{
6/voilà conp	3/la det	5/nuit subc	5/noire adjq	6/tombe verb	6/aussi cocs
6/voilà pres	3/la pper	5/nuit verb	5/noire subc	6/tombe subc	6/aussi adv
}	}	}	}	}	}

Stratégie :

1. Si la phrase ne contient aucun mot inconnu et pour un rédacteur connu, on peut utiliser une méthode classique tel que l'algorithme de Viterbi.

2. Dans le cas contraire notre objectif n'est plus de désambiguïser l'ensemble mais d'obtenir des indices permettant un intervalle de confiance
   - Si une phrase est complètement ambiguë, on commence par appliquer le vecteur de probabilité de début de phrase.
   - Si cette méthode ne donne rien, on peut faire un calcul de probabilité des chemins sur deux transitions en début de phrase ou à défaut en fin de phrase.
   - Si cela ne donne rien encore, le filtre va rejeter la totalité de la phrase comme étant insoluble. Cette situation ne s'est jamais rencontrée sur la totalité du corpus.
   - Dans ce cas CELINE devra essayer de solutionner au moins un ventre. Par exemple en effectuant des analyses syntaxiques sur la totalité de la phrase ou à défaut sur une partie de la phrase (début de phrase de préférence).

5.5.3.3 Algorithmes à résolutions locales

Le corpus d'échantillonnage étant de taille insuffisante, les matrices sont "trop" creuses et toutes les transitions ne sont pas observées. Le filtre (aussi bien d'ordre un que d'ordre deux) n'arrive pas à lever toutes les ambiguïtés.

Une conséquence pratique du problème ci-dessus est que l'on ne peut pas traiter le texte en n'avançant que lorsqu'une ambiguïté est résolue. Cela exclut les algorithmes traditionnels exposés précédemment. Nous avons développé des algorithmes à résolutions locales partout où cela est possible. Par des parcours successifs et l'utilisation séquentielle du filtrage direct ou inverse d'ordre un, d'ordre deux et du calcul de probabilité des chemins, ou par coopération avec d'autres agents du système, les mots de la phrase sont peu à peu désambiguïsés.

Pour augmenter les capacités de résolution, les sens direct et inverse de résolution sont systématiquement introduits. Par exemple, pour l'évaluation d'un enchaînement cat1 cat2 cat3, la transition cat1 cat2 peut ne pas figurer dans la matrice alors que la transition cat2 cat3 sera trouvée et permettra la sélection de cat2.
Ce changement de sens de filtrage sert également à lever des ambiguïtés que le sens direct donne comme équiprobables ou bien concurrentes au sens des intervalles de confiance.

CELINE utilise différentes stratégies selon le niveau d'ambiguïté rencontré. En ce qui concerne la correction-détection d'erreur, le filtrage automatique est basé principalement sur les chaînes de MARKOV d'ordre 1 de sens direct ou rétrograde. Les autres solutions de levée d'ambiguïté ne sont utilisées qu'en cas d'échec du filtrage d'ordre un.

1) Si la phrase est "complètement" ambiguë :
    1.1) Application du vecteur de début de phrase et parcours dans le sens direct.
    1.2) Si échecs du 1.1 alors rejet de la phrase.
2) Si une ambiguïté est présente en début de phrase ou si une ambiguïté ne peut pas être solutionnée en cours de parcours de la phrase :
- saut jusqu'à un mot non ambigu, s'il existe, et à partir de là :
- filtrage de sens inverse vers le début de phrase,
- filtrage de sens direct vers la fin de phrase.

3) Partout où cela est possible, résolution dans le sens direct de préférence au sens inverse.
4) Si le filtrage d'ordre un ne résout pas une ambiguïté, on applique un filtrage d'ordre deux direct ou inverse.
5) La stratégie consiste à boucler sur l'ordre un et l'ordre deux puis à appliquer le calcul de la probabilité des chemins et à reboucler au début. On arrête de "boucler" si l'exécution d'une boucle n'introduit aucun changement.
6) Si une ambiguïté ne peut être résolue par le filtrage statistique, on introduit une catégorie PROB (problème) que CELINE essayera de résoudre par l'étude du contexte.

5.5.3.4 Simulation de l'évolution de la désambiguïsation

5.5.3.4.1.1 Structure d'une phrase analysée, non complètement ambiguë

Nous allons supposer que sur la phrase le k^{ième} mot est supposé non ambigu.

	mot[1]	mot[2]	. . .	mot[k]	. . .	sep h	Mots de la phrase repérés par un indice i

	{	{	. . .		. . .		Début d'ambiguïté
ambiguïté [i][j]	cat1 cat2 cat3 cat4	cat6 cat7 cat8	. . .	cat_u	. . .		Les différentes catégories possibles pour le mot
	}	}	. . .		. . .		Fin d'ambiguïté
nb_ambiguïté_ par_mot[i]	4	3	. . .	1	. . .		Nombre de catégories pour le mot d'indice i
catmot[i]			. . .	1	. . .	resp	La catégorie réelle est désignée par l'indice j de l'ambiguïté[i][j] du mot d'indice i
Nombre de mots de la phrase : nbmot (1 < = i < = nbmot)							

Après l'affectation de la phrase le vecteur catmot est rempli :

- avec des zéros pour les mots ambigus,
- avec des uns pour les mots non ambigus.

Résoudre une ambiguïté consiste à remplacer le zéro par le numéro j de ambiguïté[i][j] correspondant à la catégorie morphologique la plus probable.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

(Considérons une phrase comportant 16 mots : l'indice i de ambiguïté[i][j] varie de 1 à16)

1	0	0	0	0	**1**	0	0	0	0	0	**1**	0	0	0	0

*(Initialisation : les "**1**" correspondent aux mots non ambigus : 1er mot, 6 ^{ième} mot etc.)*
(la valeur de l'indice j=1 pointe sur la catégorie unique du mot i correspondant)

5.5.3.4.1.2 Traitement fictif

Première étape : filtrage d'ordre un si possible direct et sinon de sens inverse															
1	**3**	0	0	**2**	1	4	2	0	0	5	1	3	3	2	0

(La transition directe entre les mots 1 et 2 a permis d'affecter au mot 2 la 3^{ième} catégorie proposée par l'analyse morphologique pour ce 2 ^{ième} mot : catmot[2]:=ambiguïté[2][3]).
(La transition inverse entre les mots 5 et 6 a permis d'affecter au mot 5 la 2 ^{ième} catégorie proposée par l'analyse morphologique pour ce 5 ^{ième} mot : catmot[5]:=ambiguïté[5][2]).
(etc. ...).
(Par contre la transition directe d'ordre 1 entre les mots 2 et 3 n'a pas permis d'affecter au mot 3 une des catégories proposées par l'analyse morphologique pour ce 3 ^{ième}mot.)
(etc.).

Deuxième étape :filtrage d'ordre deux si possible direct et sinon de sens inverse															
1	3	**1**	0	0	1	4	2	4	2	5	1	3	3	2	0

(Cette fois, la transition directe d'ordre 2 entre les mots 1,2 et 3 a permis d'affecter au mot 3 la 1ère catégorie proposée par l'analyse morphologique pour ce 3 ^{ième}mot.)
(etc.).

Si la deuxième étape amène un changement, on reboucle sur la première

Troisième étape
S'il reste encore des ambiguïtés on cherche à les résoudre par le calcul de la probabilité des chemins

1	3	1	**0**	2	1	4	2	4	2	5	1	3	3	2	**0**

Si la troisième étape amène un changement, on reboucle sur la première

Quatrième étape
On affecte la catégorie PROB aux ambiguïtés non encore résolues.

*(En réalité, la catégorie PROB est désignée par un indice **0** résiduel)*

5.5.3.5 Algorithme de filtrage d'ordre 1 de CELINE

Le filtre étant d'ordre 1, les ventres sont indépendants du point de vue du calcul de probabilités. La mémoire du modèle ne dépend que de la dernière catégorie rencontrée c'est-à-dire du nœud précédent. On va donc réaliser un filtrage "ventre par ventre".

L'algorithme peut s'appliquer à un corpus entier, à une phrase ou à un segment de phrase.

```
Algorithme filtrage
début
        positionner_la_première_zone {traitement zone par zone};
        affecter(zone_précédente)

        tant que (zone <= zone_finale) faire
        début
                si (zone=noeud) alors
                        traiter_noeud(zone)
                sinon
                        Si zone-précédent = nœud alors traiter_ventre(zone);
                fsi;
                zone_suivante;
                affecter(zone_précédente)
        fin;
fin.

avec

Algorithme traiter_noeud(zone)
début
        zone_précédente := nœud
        recopier_la_zone_telle_qu'elle_est(zone);
        affecter les coefficients de validité
fin;

Algorithme traiter_ventre_automatique(zone)
début
        dénombrer_les_solutions_possibles;

        calculer_la_probabilité_de_chaque_solution;

        classer_les_solutions_par_ordre_décroissant_de_probabilité;
```

```
        calculer_l'intervalle_de_confiance_de_la_solution_la_plus_probable(Ip max);

        Si ⌉solutions_concurentes
                choisir_la_solution_la_plus_probable;
                recopier_la_solution_la_plus_probable
                affecter les coefficients de validité
                zone_précédente := nœud
        sinon
                signaler_les_solutions_concurrentes;
                zone_précédente := ventre


        sauvegarder_les_résultats {sorties};
fin;
```

5.5.4 ALGORITHME DE CHANGEMENT AUTOMATIQUE DE STRATÉGIE ET COOPÉRATION AVEC LES AUTRES AGENTS

5.5.4.1 Adaptabilité du filtrage

Les trois procédures filtrage_d'ordre_un (U), filtrage_d'ordre_deux (D) et calcul_de_la probabilité_du chemin (C), appliquées localement, sont indépendantes les unes des autres. On peut donc dans l'absolu choisir 18 stratégies différentes :

[U D C] ou	[U D] C ou	U [D C]	Les crochets indiquent les boucles à exécuter tant qu'il y a des changements.
[C U D] ou	[C U] D ou	C [U D]	
[D C U] ou	[D C] U ou	D [C U]	
[U C D] ou	[U C] D ou	U [C D]	
[D U C] ou	[D U] C ou	D [U C]	
[C D U] ou	[C D] U ou	C [D U]	

Nous avons choisi pour l'instant l'ordre [UD]C ou [DU]C ainsi qu'une commutation automatique entre ces deux ordres (par exemple pour un rédacteur inconnu).

- **Pour l'écrit ou un oral de bonne qualité et les rédacteurs classés "experts", on choisit l'ordre [DU]C qui privilégie l'ordre deux.**
- Pour les textes entachés d'erreurs, on choisit l'ordre [UD]C qui privilégie l'ordre un.
- Commutation automatique, entre ces deux ordres selon un critère faisant intervenir le pourcentage des transitions résolues.

  Par exemple sur N mots, on a
  N-1 transitions d'ordre un possibles dont x résolues,
  N-2 transitions d'ordre deux possibles dont y résolues.
  Taux_ordre_un = x/(N-1)
  Taux_ordre_deux = y/(N-2)
  Si Taux_ordre_un >> Taux_ordre_deux alors ordre [UD]C sinon ordre [DU]C

5.5.4.2 Mise en œuvre

On utilise deux variables booléennes :

- changement1 : mise à vrai dans la procédure filtrage_par_calcul_du_chemin si le calcul de la probabilité des chemins affecte une catégorie morphologique à un mot.

- changement2 : mise à vrai dans les procédures filtrage_ordre_un et filtrage_ordre_deux si le filtrage markovien d'ordre un ou deux affecte une catégorie morphologique à un mot.

La procédure evaluation_gestion_situation pilote le fonctionnement du filtrage Markovien et peut exécuter :

1. Un verrouillage de la stratégie en ne changeant pas le choix de l'ordre.
2. Une simple commutation de l'ordre (choix_ordre := (choix_ordre +1) mod 2)
3. Mais elle peut aussi commander directement ou indirectement
   - l'affectation de la catégorie PROB à une forme textuelle inconnue.
   - un travail de coopération aux autres agents de CELINE (notamment l'analyseur syntaxique)
   - une pause du traitement (pour laisser le temps aux autres agents d'intervenir)
   - un arrêt du traitement en cas de situation d'échec total
   - une modification des seuils applicables aux divers ordres

Cette procédure evaluation_gestion_situation est implantée dans un agent « intelligent » (agent PILMARK de désambiguïsation) dirigeant les travaux d'autres agents réactifs (agent de traitement statistique des modèles de Markov).

```
procédure filtrage_statistique( choix_ordre : entier ; commut_auto : boolean);
début
        changement1:=vrai;
        tant_que changement1 faire
          début
                changement1:=faux;
                changement2:=vrai;
                tant_que changement2 faire
                        début
                                changement2:=faux;
                                cas choix_ordre dans
                                        0 :     début
                                                        filtrage_ordre_un;
                                                        filtrage_ordre_deux;
                                                        evaluation_gestion _situation;
                                                fin;
                                        1 :     début
                                                        filtrage_ordre_deux;
                                                        filtrage_ordre_un;
                                                        evaluation_gestion _situation;
                                                fin;
                                fin_cas;
                                si commut_auto alors
                                        début
                                          calcul de taux ordre un ;
                                          calcul de taux ordre 2
    si taux ordre 1 > taux ordre 2      alors choix_ordre = 1 sinon
                                                                choix ordre = 2
                                          evaluation_gestion _situation;
                                        fin
                        fin;
                filtrage_par_calcul_du_chemin;
                evaluation_gestion _situation;
```

```
            fin;
        écrire_résultat-dans_fichier_de_sortie;
fin;
```

5.6 CONCLUSION

Les modèles de Markov permettent l'introduction d'une composante quantitative fondamentale dans les prises de décision relatives aux ambiguïtés des catégories morphologiques. Cet apport déjà intéressant pour l'analyse d'un texte non entaché d'erreur voit son importance multipliée lorsqu'il s'agit de proposer des solutions pour une forme textuelle inconnue supposée être une erreur.

Les modèles de Markov vont aussi intervenir dans le calcul des poids des phrases concurrentes, poids permettant un choix dans un mode automatique ou un classement dans un mode interactif.

Ce chapitre a permis de présenter les possibilités de calcul de l'agent de proposition statistique de catégories morphologiques ainsi que la stratégie mise en œuvre dans l'agent de désambiguïsation lexicale PILMARK

6. UN MODÈLE LINGUISTIQUE PARTIEL SUFFISANT

Finalité

Dans les chapitres précédents, nous avons vu se préciser peu à peu quelles spécificités du rédacteur devaient être prises en compte en vue d'une optimisation et d'une plus grande fiabilité de la correction des erreurs. L'introduction (chapitre I), les heuristiques de correction (chapitre II), le modèle du maître d'école (chapitre IV), les modèles de Markov (chapitre V) nous ont permis d'en approcher certains aspects sous des angles éventuellement différents en les approfondissant plus ou moins.

Ces points épars demandent une synthèse et une poursuite de la réflexion sur deux plans distincts :

- celui de la connaissance du rédacteur, finalité de ce chapitre,
- celui de l'architecture d'un système permettant l'apprentissage et la prise en compte de cette connaissance (présenté dans le chapitre sur CELINE).

Initialement le Modèle Linguistique Partiel Suffisant d'un rédacteur se composait des seuls éléments linguistiques de ce rédacteur sous la forme :

- d'un lexique,
- d'un ensemble de règles de grammaire.

Actuellement, le MLPS regroupe l'ensemble des données spécifiques nécessaires à la correction des erreurs pour un rédacteur particulier : les différents taux déjà rencontrés, les matrices de transitions des modèles de Markov, les règles de correction de ces modèles etc.

Résumé

Nous commencerons par nous intéresser à la reconnaissance lexicale, lourd problème compte tenu de la variété des domaines possibles.

Dans les fragments de protocole de travail sur la structure de données, nous avons décidé que dès les premiers mots reconnus, les analyseurs syntaxiques commençaient à rechercher les structures terminales. La validation d'une phrase finale va permettre de mémoriser les différentes structures composantes de l'arbre syntaxique. La deuxième partie du chapitre va donc s'intéresser à la détermination en extension de la grammaire du rédacteur.

En troisième partie, les modèles de Markov complèteront cette définition de la grammaire en permettant de calculer des poids pour les diverses règles.

Nos conclusions sur la non convergence des matrices observée au chapitre 5 nous permettront alors d'introduire des corrections de ces modèles sous la forme de règles contextuelles de corrections.

Nous dresserons ensuite une liste récapitulative de l'ensemble des coefficients, seuils et taux divers par rédacteur.

En conclusion, cette définition du rédacteur va nous permettre d'introduire une adaptation de l'architecture du système qui, pour un utilisateur particulier, va se simplifier en constituant *une image personnalisée du système de correction*.

6.1 ASPECTS LEXICAUX DU MODÈLE LINGUISTIQUE PARTIEL SUFFISANT

6.1.1 APPRENTISSAGE DES DOMAINES D'UN UTILISATEUR

6.1.1.1 Les agents lexicaux

Le pilote des activités lexicales PAL (Menézo 96 b) va distribuer la tâche d'identifier chacun des mots (et de fournir racines, catégories et variables morphologiques) aux différents agents du secteur d'activités que nous appellerons *agents lexicaux*. Ce pilote pourra également intervenir lors d'une étape de génération après une correction.

Ces agents peuvent être éligibles ou pas selon le paramétrage du système, paramétrage défini par l'utilisateur selon le(s) domaine(s) du texte traité et le(s) domaine(s) de leurs bases de connaissances. Rappelons que notre système se voulant « universel » pourrait accéder à des centaines voir des milliers de domaines.

Pour certains domaines très spécialisés, le pilote PAL pourrait s'adresser à un agent monolithique indépendant assurant l'ensemble des traitements (reconnaissance lexicale, analyse syntaxique, analyse sémantique) et fournissant une réponse finale sur la phrase ou le paragraphe envisagé.
Exemple : Les serveurs canadiens sur la météorologie qui corrigent les bulletins météo dans une soixantaine de langues (traductions à partir de bulletins en langue anglaise utilisée par exemple au J.O.d'Atlanta).

Le coefficient d'utilisation, rencontré au chapitre 4, va permettre à tout instant de déterminer, pour une forme textuelle à identifier, l'ordre optimal des lexiques. Ce coefficient va être pris en compte par le maître d'école et nous verrons qu'il dépend :

1. du rédacteur,
2. pour un rédacteur, du texte,
3. pour un texte, d'un rédacteur de la zone de texte traitée.

La stratégie d'accès aux différents lexiques utilise cinq éléments :

- La liste des cinq derniers lexiques accédés avec succès,
- les coefficients d'utilisation,
- une modélisation de Markov des couples (mots/lexiques) ou des successions de lexiques,

et en vue des relectures :

- des feuilles lexico-syntaxiques,
- une mémorisation mot par mot du lexique optimum.

6.1.1.2 Liste des derniers lexiques utilisés

Cette solution de gestion peu coûteuse est proposée par la majorité des systèmes d'analyse multi-lexiques.
Avantage : simplicité de mise en œuvre.
Inconvénients : la liste est souvent limitée à 5 ou 10 lexiques et, en général, ne contient pas d'ordonnancement des accès.

Nous préférons à cette solution la gestion de listes non bornées avec des probabilités d'utilisation : les coefficients d'utilisation.

6.1.1.3 Les coefficients d'utilisation par session

6.1.1.3.1 Généralités

Le coefficient d'utilisation de l'agent lexique n°i représente la probabilité a priori pour qu'un mot quelconque soit reconnu par le lexique n°i. Ce coefficient contribue à déterminer l'ordre de recherche dans les lexiques pour l'identification d'une forme textuelle.
Nous appellerons *session* le traitement d'une partie plus ou moins longue de texte(s) et nous désignerons une session par une lettre F, S, T, U, G.
Selon la session envisagée, les **C**oefficients d'**U**tilisation vont être nommés respectivement CU**F**, CU**S,** CU**T**, CU**U**, CU**G**. Nous utiliserons ces abréviations par la suite.

Liste des types de sessions		
F	**F**enêtre sur le texte.	Cette fenêtre constitue un échantillon de l'ordre de la centaine de mots soit environ un tiers de page ou une demi-page.
S	**S**ection	Cette section va correspondre à une partie du texte (paragraphes, rencontre d'un titre etc.**)**
T	**T**exte entier.	Par exemple, un chapitre de thèse, un article, une lettre.
U	**U**tilisateur	La session est l'ensemble des textes de l'utilisateur.
G	**G**énéral	Il s'agit cette fois de l'ensemble des textes examinés par le système.

6.1.1.3.2 Principe de calcul du coefficient d'utilisation

Pour chaque session on mémorise le nombre de mots reconnus dans chaque lexique :

> N représente l'ancien nombre total de mots, N' le nouveau après la session.
> On a N'=N+n1+n2+n3+...+nn.

N° du lexique	Nombre de mots reconnus	Probabilité Ancienne	Nouvelle probabilité
1	n1	p1	(p1*N+n1)/N'
2	n2	p2	(p2*N+n2)/N'
3	n3	p3	(p3*N+n3)/N'
...	...	...	...
n	nn	pn	(pn*N+nn)/N'
Vérification de la cohérence par la somme des nouvelles probabilités : [(p1+p2+p3+...+pn)*N+n1+n2+n3+...+nn]/N' =(1*N+n1+n2+n3+...+nn)/N'=N'/N'=1			

La mise à jour des fréquences est faite par le maître d'école (agent MAEC) à chaque fin d'analyse d'une phrase. Nous avons vu au chapitre II que le tableau structure de données contenait les marques de validité pour chaque forme textuelle et que celles-ci mémorisaient l'agent à l'origine de la validation. Il est donc facile par simple relevé de faire la mise à jour nécessaire.

6.1.1.3.3 Coefficients d'utilisation par fenêtre

Pour illustrer l'aspect dynamique (en caricaturant « en temps réel ») du suivi du texte, nous garderons en mémoire l'exemple d'un chercheur spécialiste de chimie organique travaillant dans le domaine de l'agriculture et écrivant un article comportant dans

l'ordre : une introduction, un historique, un développement technique (avec inclusion d'expressions météorologiques), une conclusion et pour terminer une bibliographie.

Les coefficients d'utilisation vont permettre une optimisation par une gestion dynamique de la planification des recherches dans les différents lexiques. Les coefficients vont être obtenus à partir d'un échantillon du texte (la fenêtre).

Diverses stratégies sont utilisables :

1. La plus souple (inévitable pour un système sans expérience) : les CUF sont tous initialisés à zéro (refus d'idées préconçues sur le type de texte) tous les X mots (pour fixer les idées, X de l'ordre de la centaine). Dans l'exemple, si on passe sur la partie historique ou sur la bibliographie, le coefficient va montrer une utilisation élevée du lexique des noms propres alors que sur le développement technique, le coefficient montrera une utilisation fréquente du lexique de la chimie organique ou de celui de la météorologie.
2. La plus efficace en moyenne pour un système avec expérience : initialiser les CUF avec des valeurs moyennes (cf. coefficient d'utilisation par utilisateur CUU et à défaut coefficient d'utilisation général CUG).

6.1.1.3.4 Coefficients d'utilisation par section

Les CUF entraînent un « typage » local du texte. On peut envisager (tous les X mots ou à chaque changement de titre pour un texte avec plan) de mémoriser la valeur des coefficients.

Lors d'une reprise du texte, la récupération de ces coefficients va permettre d'obtenir une certaine optimisation immédiate sans attendre une convergence à partir de valeurs arbitraires.
Dans le cas du chercheur en chimie, il est probable que tous les paragraphes d'un même niveau de différents articles vont montrer des utilisations des lexiques comparables (par exemple pour l'introduction, pour la conclusion, pour la bibliographie).

La mémorisation des CUF pourrait s'intégrer dans *des feuilles lexico-syntaxiques* comparables, en vue de la correction des erreurs, aux feuilles de styles pour la mise en forme d'un texte. Ce typage du texte sera aussi utilisé au niveau syntaxique.

6.1.1.3.5 Coefficients d'utilisation par texte

Les principes de calcul restent les mêmes mais cette fois le cumul des reconnaissances par lexique est fait au niveau du texte entier.
La mémorisation de ces coefficients va permettre lors d'une *reprise du même texte* d'initialiser les CUF à des valeurs non arbitraires et de converger plus vite vers des valeurs efficaces. Pour comprendre l'utilité de cette possibilité, il faut se rappeler le cadre du système avec de très nombreux agents lexicaux accessibles.
Le coefficient d'utilisation par texte représente pour un lexique la moyenne des coefficients par fenêtre de ce lexique pour ce texte. Ces coefficients représentent aussi une forme d'acquisition d'une expérience par le système.

6.1.1.3.6 Coefficients d'utilisation par utilisateur

La mémorisation de ces coefficients va permettre, lors du traitement *d'un nouveau texte*, d'initialiser les CUF à des valeurs non arbitraires en permettant en moyenne une optimisation.
Le coefficient d'utilisation par utilisateur représente pour un lexique la moyenne des

coefficients par texte de ce lexique pour cet utilisateur.

6.1.1.3.7 Coefficients d'utilisation généraux moyens

Il va permettre une initialisation optimisée cette fois pour un *nouvel utilisateur*. Par exemple, imaginons un système de correction utilisé par les chercheurs d'une même équipe. Le système a acquis une expérience et une connaissance de ces chercheurs et va donc «réagir intelligemment » face à un nouvel utilisateur.
Le coefficient d'utilisation général moyen représente pour un lexique la moyenne des coefficients par utilisateur pour ce lexique.

6.1.1.3.8 Relecture d'un texte et mémorisation par mots

En descendant vers une granularité plus fine, il est possible de mémoriser le lexique optimum pour chaque mot. Dans le cadre d'un logiciel moderne, cela ne semble pas plus exigeant que la possibilité de mémoriser les marques de formats du style (gras, italique, souligné, police, taille, ...) pour chaque caractère.

Cette solution est très intéressante car, en vue des relectures d'un même texte, l'important travail de recherche accompli par exemple pour un mot technique d'un domaine très particulier ne sera pas à refaire. Rappelons que le travail, dans ce cas, consiste à trouver le ou les lexiques techniques (parmi plusieurs centaines d'autres) contenant ce mot avec éventuellement une désambiguïsation sémantique si le mot appartient à plusieurs domaines avec des sens différents (exemple la *fréquence* des physiciens, nombre de fois où un phénomène périodique se produit par seconde, exprimée en Hertz, et la *fréquence* des statisticiens égale au nombre d'occurrences d'un fait).

Nous pouvons aussi généraliser les informations à mémoriser pour chaque mot :

- le lexique ayant permis le reconnaissance du mot,
- la catégorie morphologique, affectée,
- la variable morphologique *genre,*
- la variable morphologique *nombre,*
- Pour un verbe : les variables *mode, temps, personne*

Pour diminuer la taille des informations mémorisées, les diverses informations utiles sont codées sous la forme de caractères ASCII avec une interprétation binaire et sont précédées et suivies de caractères d'exception.

Codage du numéro du lexique :

> valeur binaire du numéro sur 8 positions soit de 0000 0000 à 1111 $1111|_2 = 255|_{10}$ soit 256 lexiques possibles.

Codage de la variable morphologique possible :

> numéro de 0 à 127 codé sur 4 positions binaires.

Le genre et le nombre sont codés sur deux bits. Ce code respecte les valeurs exposées pour la qualité dans la méthode des structures (chapitre VII) si on interprète le premier bit comme un bit de signe (0 positif et 1 négatif) :

Codage du genre sur deux bits			
Codage	Signification	Exemples	Structures
0 0	pas de genre	*et*	0 * 0 = 0
1 1	féminin	*chienne*	-1*1 = -1
0 1	masculin	*chien*	+1 * 1 = +1
1 0	masculin et féminin	*jaune*	-1 * 0 = 0

Codage du nombre sur deux bits			
Codage	Signification	Exemples	Structures
0 0	pas de nombre	*or*	0 * 0 = 0
1 1	singulier	*chien*	-1 * 1 = -1
0 1	pluriel	*chiens*	+1 * 1 = + 1
1 0	Singulier et pluriel	*mois*	-1 * 0 = 0

Récapitulons :

Le codage de l'accès lexical demande 1 octet.

Le codage de la catégorie morphologique et des variables morphologiques classiques demandent 1 octet.

Comme nous allons de plus mémoriser des informations syntaxiques ou sémantiques, pour les besoins du calcul ci-dessous nous supposerons qu'au total, il nous faut 8 octets par mots.

Examinons l'accroissement de taille d'un fichier mémorisant pour chaque mot cet ensemble d'informations :

Sur un texte particulier (un document interne de synthèse d'avancement de la thèse de 32 pages ; tous schémas enlevés) WORD6 nous indique :

Nombre de mots : 13 504.
Nombre de caractères : 74 919.
Taille du fichier : 322 048 octets.

Si nous rajoutons 8 octets à chaque mot nous obtenons alors 13 504 * 8 = 108 032 octets supplémentaires soit un total de 322 048 + 108 032 = 430 080 octets pour le fichier.

Nous pouvons calculer un taux « volumique » de codage des informations supplémentaires :

Informations Word 6	(322 048 - 74 919) / 74 019	3,34
Informations lexico-syntaxiques	108 032 / 74 019	1,45
Informations totales	(430 080 -74 919) / 74 919	4,74

Nous observons certes un accroissement de la taille du fichier mais cet accroissement semble raisonnable et acceptable puisque d'environ 50 % (1,45 / 3.34 = 0,43) de ce qui est utile à WORD6 pour mémoriser le style, la mise en page etc.

Cette mémorisation par mot, si elle représente un progrès, en coût pour les relectures et des travaux successifs sur un même texte, a l'inconvénient d'être volatile en cas de changement de texte. D'où l'idée de mémoriser les informations dans une table persistante : la table lexicale du MPLS.

En réalité ces deux solutions vont être complémentaires :

- Face à un nouveau texte le MPLS va amener un gain en coût de recherche très important.
- Pour un texte déjà traité par contre, la mémorisation mot par mot semble fondamentale.

CELINE va intégrer ces solutions dans sa stratégie de travail

6.1.2 TABLE LEXICALE DU MPLS

Le nombre de mots utilisés par un rédacteur étant en général très faible, il peut être envisageable au moins, pour la majorité des utilisateurs, de dresser dans une table-lexique la liste en extension des mots utilisés avec les références des lexiques correspondants ainsi que les principaux traits morphologiques.

Exemple de table-lexique d'un rédacteur							
Alphagramme	Graphie	Catégorie Morphologique	Genre	Nombre	Personne (verbe)	Mode / Temps (V)	Lexique de référence
	...						
achelv	cheval	subc	mas	sin			PILAF
acheuvx	chevaux	subc	mas	plu			PILAF
	...						

Le temps de recherche sur une table de ce type comportant quelques milliers de mots est insignifiant.

En présence d'un mot inconnu, avant de mettre en œuvre le traitement sous-jacent au système CELINE complet, une proposition de correction va être recherchée à partir de cette table en utilisant des méthodes peu coûteuses (par exemple clé squelette ou traitement alphabétique complet).

6.1.3 MODÉLISATION DE MARKOV DES ACCÈS LEXICAUX DU RÉDACTEUR

Les modèles de Markov donnent souvent de bons résultats dans des applications de natures très différentes. Nous avons donc envisagé leur application à l'ordre d'accès aux différents lexiques.

Hypothèse 1 :
L'agent PSCM permet de prévoir pour un mot les catégories morphologiques concurrentes à partir des catégories des mots qui le précèdent. On peut donc en déduire une liste de lexiques possibles pour certain(s) mot(s) à traiter et principalement quand un outil lexical leur est consacré(s). Cette méthode (utilisation d'un modèle tri-classe) semble efficace même en l'absence de désambiguïsation car elle réduit fortement le nombre de lexiques concurrents.

> Exemple :
> On peut imaginer que dans un domaine particulier, tel que par exemple le domaine juridique, un agent particulier V soit spécialiste des verbes du domaine. Si les catégories morphologiques des mots précédents, déjà reconnus, laisse prévoir que la catégorie suivant sera un verbe, on peut placer l'agent V en tête des agents éligibles.

Hypothèse 2 :
Les textes contiennent souvent des structures qui se répètent.

Exemple :
Chez Dupond (Dupond 78) (Dupond 84) la théorie de … est …
Au contraire, chez Durand (Durand 90) l'idée qui prévaut est …

On peut donc aussi envisager de travailler directement sur des chaînes de Markov des lexiques utilisés, c'est-à-dire l'enchaînement des lexiques.

Ces deux hypothèses restent à vérifier. Elles peuvent conduire à des heuristiques qui interviendraient ponctuellement pour l'ordre des accès lexicaux en face de certaines parties de texte ou sur la totalité d'un texte mono-domaine.

6.2 MODÉLES DE MARKOV

Pour mémoire, nous mentionnons les modèles qui font partie du MPLS et que nous avons détaillés au chapitre IV. Le système va mémoriser les matrices de transitions en profitant de leur aspect creux pour diminuer leur encombrement.

Taille des matrice de transitions avec 61 catégories morphologiques :

- pour l'ordre un, on peut utiliser des matrices carrées 61 x 61 ce qui représente par exemple pour les "N_{ij}" déclarés comme "integer" (2 octets) un espace mémoire de 2 x 61 x 61 = 7442 octets soit environ 7,3 KO.
- pour l'ordre deux, une matrice de même type à trois dimensions représente un encombrement mémoire de 2 x 61^3 = 453962 octets soit environ 443 KO.

Une réduction de la taille des fichiers matrices est possible en prenant en compte l'aspect creux des matrices d'ordre un et deux :

Exemple : résultats d'apprentissage sur le fichier " MURIP.fca " :

Nombre de catégories morphologiques	: 61
Nombre de substitutions effectuées	: 4
Nombre *théorique* de types de transitions d'ordre 1	: 3249
Nombre *théorique* de types de transitions d'ordre 2	: 185193
Nombre de types de transitions d'ordre un *observées* :	163
Nombre théorique de types de transitions d'ordre 2 compte tenu du nombre réel de transitions d'ordre 1	: 9291
Nombre de types de transitions d'ordre deux *observées*	: 303

Les pourcentages de remplissage des matrices sont donc très faibles :

Pour la matrice d'ordre un 163/3249 = 0,05
Pour la matrice d'ordre deux 303/185193 = 1,64 x10^{-3}.

Solutions finalement retenues :

Matrice carrée 61x61 pour la matrice d'ordre un. Soit N le nombre d'éléments non nuls de cette matrice.
Matrice rectangulaire 61 x N pour la matrice d'ordre deux. Dans le cas de l'exemple ci-dessus nous obtenons un encombrement mémoire de 2x61x163 octets soit environ 19,4 KO.

Total des deux matrices : 26,7 KO

6.3 ASPECTS GRAMMATICAUX DU MODÈLE LINGUISTIQUE PARTIEL SUFFISANT

Martin Rajman a présenté dans TAL (Rajman 95), un panorama complet de l'approche probabiliste de l'analyse syntaxique au niveau des concepts généraux, au niveau de la probabilisation des modèles non contextuels (grammaires stochastiques) et des modèles à dépendances contextuelles. Aucune des nombreuses solutions proposées ne correspond exactement à notre double contrainte : conserver un potentiel de couverture maximale et s'adapter aux besoins précis d'un utilisateur.

Nous allons envisager deux constructions possibles du modèle grammatical :

- une construction par le calcul,
- une construction statistique.

6.3.1 CONSTRUCTION PAR CALCUL

6.3.1.1 Principe

Notre voie de recherche consiste à prendre une grammaire (par exemple de constituants) la plus générale possible et de construire l'ensemble des dérivations possibles avec le vocabulaire non terminal tout en ne conservant que les dérivations de probabilité supérieure ou égale à un certain seuil. Les probabilités des chaînes sont calculées en utilisant les matrices de transitions de chaînes de Markov portant sur un modèle bi-classes des catégories morphologiques.

6.3.1.2 Construction de l'ensemble des règles

Considérons un utilisateur "n'entassant jamais" plus de deux adjectifs devant et derrière le substantif avec éventuellement des conjonctions de coordination entre les adjectifs soit 8 mots au maximum (hypothèse raisonnable !). La matrice de transition d'ordre 1 permet de calculer la probabilité de chaque enchaînement.

Nombre de mots de la structure	Nombre de solutions	Enchaînements en fonction du nombre de mots de la structure
1		Sans solution
2	1	det subc
3	2	det adjq subc det subc adjq
4	3	det adjq adjq subc det adjq subc adjq det subc adjq adjq
5	4	det adjq conj adjq subc det adjq adjq subc adjq det adjq subc adjq adjq det subc adjq conj adjq
6	4	det adjq conj adjq subc adjq det adjq conj adjq subc adjq det adjq adjq subc adjq adjq det adjq subc adjq conj adjq
7	2	det adjq adjq subc adjq conj adjq det adjq conj adjq subc adjq adjq

8	1	det adjq conj adjq subc adjq conj adjq

Nous pouvons déterminer des règles de définition d'un groupe nominal engendrant les enchaînements du tableau précédent :

Gn	→	Gn Conj Gn
Gn	→	Det Ga Subc Ga
Ga	→	Ga conj Ga
Ga	→	Adjq
Ga	→	Adjq Conj Adjq
Ga	→	Nil

Le nombre de dérivations engendrées par ces simples règles est infini.

Pour notre rédacteur, nous voyons alors qu'à la place d'un nombre infini de solutions nous avons un maximum de 17 possibilités. De plus il est facile de réduire ce nombre :

- Si nous faisons intervenir le nombre de mots de la structure, obtenu par comptage, nous avons au plus quatre motifs possibles.
- Nous pouvons aussi faire intervenir la probabilité des enchaînements. Par exemple dans le cas du texte *murip*, si nous appliquons un seuil de probabilité de 0,4 il nous reste alors 3 solutions possibles.

Avantage de cette méthode :
Elle est directement applicable même pour un rédacteur inconnu en considérant un ensemble de règles et une matrice de transitions standards.

Inconvénients de cette méthode :
Cette méthode repose sur l'écriture de la grammaire et les modèles de Markov. Elle cumule donc les inconvénients des deux. En particulier se pose le problème de la stabilité de cette matrice.

6.3.2 APPRENTISSAGE PAR LE MAITRE D'ÉCOLE

Lorsqu'une phrase est validée par le rédacteur en mode interactif ou par le système en mode automatique, le tableau structure de données contient les différentes structures correspondantes à son analyse syntaxique. On peut donc repérer les diverses règles utilisées.

Exemple :

Le beau cheval noir racé et les juments courent furieux et affamés.

Le tableau va permettre de reconstituer l'analyse syntaxique complète :

P(gn(gn(detp(le,mas,sin)adjq(beau,mas,sin)subc(cheval,mas,sin)adjq(noir,mas,sin)adjq(racé,mas,sin))coco(et)gn(detp(les,mas,fem,plu)subc(juments,fem,plu)))gv(verb(courent,plu)adv(adjq(furieux,mas,sin,plu)coco(et)ppas(affamés,mas,plu)))))

La maître d'école va pouvoir déterminer l'utilisation des règles suivantes et les porter dans l'ensemble des règles applicables :

P -> GN GV

GN	->	GN	COCO	GN		
GN	->	DET	ADJQ	SUBC	ADJQ	ADJQ
GN	->	DET	SUBC			
GV	->	VERB	GADV			
GADV	->	ADJQ	COCO	PPAS		

Ensuite par comptage de la fréquence de rencontre de chaque règle on va pouvoir déterminer une probabilité associée.

Inconvénient :
Cette méthode nécessite une période d'apprentissage pendant laquelle elle est inintéressante (reconnaissance de trop peu de structures). L'importance de cette phase d'apprentissage est une fonction croissante de la richesse linguistique du rédacteur. Comme pour l'auto-correction, un taux du nombre de structures nouvelles va pouvoir déterminer la fin de cette période d'apprentissage.

Avantage :
Obtention de règles bien adaptées (et pour cause) au rédacteur.

6.3.3 STATÉGIE PAR COMBINAISON

Tant que le rédacteur n'est pas connu, le premier jeu des règles est applicable. Pendant cette application, le deuxième jeu se construit peu à peu.

Dans tous les cas, la stabilité des matrices de transitions est un signe montrant que l'apprentissage est terminé.

Nous pouvons alors envisager la possibilité de faire une analyse locale sur quelques mots par un simple parcours linéaire : nous disposerons alors de *structures* facilement reconnaissables permettant de faire directement de la vérification correction d'accords.

6.4 CORRECTION GRAMMATICALE

6.4.1 PRINCIPE ET APPRENTISSAGE

6.4.1.1 Principe

Nous avons vu au chapitre V les limites des modèles de Markov et les erreurs que peut engendrer le filtrage Markovien. Une des causes essentielles est la taille insuffisante des échantillons.

Face à un rédacteur occasionnel et pour faire face à ce problème, les matrices du rédacteur peuvent être initialisées à partir de matrices standards définies pour un utilisateur non typé à partir de corpus de taille importante. Au fil des travaux du rédacteur on peut alors espérer tendre vers une convergence et une stabilité des matrices.

En attendant, la correction grammaticale proposée est un palliatif à la non personnalisation des matrices de transitions. Elle part de l'idée que le rédacteur suit en général les habitudes classiques d'expression mais qu'il a aussi ses propres habitudes.

Comme nous l'avons vu, ces anomalies peuvent être détectées en mode interactif et le système va en profiter pour calculer des règles d'adaptation aux matrices.

6.4.1.2 Relevé des erreurs du filtrage Markovien

L'agent maître d'école MAEC compare les enchaînements valides sur les phrases finales retenues par l'utilisateur et ce que donnerait l'application des modèles de Markov.

En vue de statistiques et d'une étude linguistique, le maître d'école édite dans un fichier la comparaison. Chaque phrase du corpus est réécrite. Le module signale les mots pour lesquels le filtrage donne un résultat erroné et indique la catégorie "fausse" (Filtre) et la catégorie "juste" (Réel) correspondante.

A intervalles réguliers (par exemple toutes les 20 phrases) les erreurs sont groupées et comptées par catégories et par contextes.

> Exemple : extrait d'un fichier d'erreurs *(avec commentaires explicatifs additionnels en italiques)* :

93/et **la** fenêtre m'indique les mesures donc je mesure seulement une partie voilà c'est bon h /

(22) **la** Filtre : pper Reel : det

*(les erreurs sont numérotées (22) le mot ambigu est désigné : "**la**"*
le filtre a choisi : pper; la catégorie réelle est : det)

32/ben on va redessiner juste **la** h /

(23) **la** Filtre : pper Reel : det

77/ce que je viens **de** sélectionner **en** sonagramme donc je rappelle "processing" h /

(24) **de** Filtre : det Reel : prep

(25) **en** Filtre : pper Reel : prep

etc. ...

RECAPITULATIF PAR TYPE D'ERREURS :

coco	(pper->det)	subc	Nb = 2 Erreurs: 4; 21

(Nb=2 fois dans le texte (erreurs numérotées 4 et 21), le filtre à choisi "pper" à la place de "det" les catégories formant le contexte étaient : coco et subc.)

apdi	(pper->det)	subc	Nb = 2 Erreurs: 7; 26
subc	(adjq->apdi)	pnp	Nb = 2 Erreurs: 8; 17
verb	(det ->pper)	infi	Nb = 3 Erreurs: 9; 15; 18
resp	(adv ->apdi)	intj	Nb = 1 Erreurs: 6

etc. ...

6.4.1.3 Proposition automatique de règles contextuelles.

Un agent GENREG (génération de règles) codifie les erreurs du filtre en tenant compte du contexte et mémorise des propositions de règles.

Les règles contextuelles de correction s'écrivent sous la forme :

> X cat1 Y ---> X cat2 Y (exemple : *verb **ppas** ppas --->verb **ppae** ppas*

ce qui signifie que la catégorie cat1 (*ppas*) doit être remplacée par la catégorie cat2 (*ppae*) à condition que cat1 soit précédée de la catégorie X (*verb*) et suivie de la catégorie Y (*ppas*). On dit que X et Y sont les deux catégories morphologiques formant le contexte d'application de la règle : cat1 se réécrit cat2.

Exemples de règles

(1)	verb	**ppas**	ppas	--->	verb	**ppae**	ppas
(2)	infi	**adv**	apdi	--->	infi	**pnp**	apdi
(3)	pnp	**adv**	tech	--->	pnp	**apdi**	tech
(4)	coco	**pper**	subc	--->	coco	**det**	subc
(5)	intj	**adv**	resp	--->	intj	**apdi**	resp
(6)	resp	**adv**	intj	--->	resp	**apdi**	intj

etc. ...

Une difficulté réside dans le fait que, parmi les règles proposées, certaines possèdent un membre de gauche identique. La probabilité affectée à chaque règle va intervenir pour calculer le poids de la solution globale.

Exemple :

(1) X cat1 Y ----> X cat1 Y p=0,25
(2) X cat1 Y ----> X cat2 Y p=0,5
(3) X cat1 Y ----> X cat3 Y p=0,25

Interprétation :

La règle (1) devrait s'appliquer une fois sur quatre.
(Dans ce cas, il s'agit d'une règle fictive puisque cat1 se réécrit en cat1).
La règle (2) devrait s'appliquer une fois sur deux.
La règle (3) devrait s'appliquer une fois sur quatre.

Nous allons maintenant présenter le calcul des probabilités associées aux règles et les critères de validations possibles.

6.4.1.4 Calcul des probabilités associées.

On considère le membre de gauche X cat1 Y des règles proposées par le module GENREG :

(1) X cat1 Y ----> X cat1 Y
(2) X cat1 Y ----> X cat2 Y
(3) X cat1 Y ----> X cat3 Y

D'une part, on dénombre les occurrences dans le corpus de l'enchaînement X cat1 Y. Le texte contenant les enchaînements exacts, on obtient une règle fictive de pseudo-réécriture correspondante aux cas où il n'y a pas lieu de changer cat1.

X cat1 Y ----> X cat1 Y Occurrence : N_1

D'autre part, on recherche le nombre d'occurrences de chaque réécriture. On obtient :

X cat1 Y ----> X cat2 Y Occurrence : N_2
X cat1 Y ----> X cat3 Y Occurrence : N_3
X cat1 Y ----> X cat4 Y Occurrence : N_4

Remarque : les N2, N3, N4 ... correspondent aux Nb des listes d'erreurs du §.7.4.1.2

* On calcule ensuite la probabilité associée à chaque règle i :

et on obtient les règles stochastiques :

$\overline{j=1}$ ^{j}

X cat1 Y ----> X cat1 Y Probabilité P_1

X cat1 Y ----> X cat2 Y Probabilité P_2
X cat1 Y ----> X cat3 Y Probabilité P_3
X cat1 Y ----> X cat4 Y Probabilité P_4

Ces règles déduites du fichier des erreurs demandent à être validées avant application.

6.4.1.5 Validation automatique d'une règle contextuelle de correction.

La validation des règles demande de choisir un critère. Diverses stratégies sont possibles :

6.4.1.5.1 Filtre « Tout ou rien »

Le filtre actuel est un filtre "en tout ou rien" : une règle n'est validée que si sa probabilité est égale à un.

X cat1 Y ----> X cat1 Y Occurrence N_1 Probabilité P_1:
X cat1 Y ----> X cat2 Y Occurrence N_2 Probabilité P_2:

1) $P_1 = 0$: on ne rencontre jamais l'enchaînement X cat1 Y dans l'échantillon après désambiguïsation

2) Règle 2 validée si $P_2 = 1$: le module GENREG propose une seule règle de réécriture pour X cat1 Y.

On peut alors réaliser une substitution systématique n'engendrant aucune nouvelle erreur lors de son exécution.

6.4.1.5.2 Filtrage avec des seuils de probabilité.

Si on considère un seuil de probabilité s la règle i n'est validée que si $P_i > s$

L'application d'une règle peut engendrer un certain nombre d'erreurs. Par exemple avec s = 0,8 et une application de la règle un très grand nombre de fois, le pourcentage des erreurs introduites pour cette correction va tendre vers 0,2 = 20%

Cet aspect négatif peut toutefois être compensé par le nombre supérieur de règles validées et donc appliquées à la correction du filtrage de Markov. Le choix d'un seuil élevé pouvant entraîner globalement plus d'erreurs corrigées que d'erreurs introduites. Le bilan peut être positif à condition de disposer là encore d'échantillons de taille suffisante.

Un agent "**GENREGLE**" permet de valider ou non une règle contextuelle proposée pour un utilisateur. Ce module crée un fichier (xxxxx.reg) des règles validées ou non avec mémorisation des probabilités associées.

Une autre étape dans la validation des règles consiste à regrouper dans un fichier unique REGLES.REG l'ensemble des règles validées sur tous les textes et donc valides sur l'ensemble du corpus

6.4.2 FILTRE GRAMMATICAL PILGRAM.

Le filtre effectue un parcours séquentiel du filtrat Markovien et pour toute occurrence des membres de gauche X cat1 Y des règles grammaticales validées, le filtre substitue à la catégorie cat1 la catégorie cat2 de la règle la plus probable, à condition que celle-ci

respecte une contrainte supplémentaire imposée par le choix d'une stratégie de filtrage. Dans le cas où la contrainte entraînerait une non application de la règle la plus probable, on considère la règle de probabilité immédiatement inférieure et ainsi de suite (rappel : toutes les règles envisageables ont une probabilité supérieure ou égale au seuil s).

Quelques stratégies possibles :

- 1) La catégorie remplaçante cat2 doit être incluse dans les solutions proposées par l'analyse morphologique pour le mot correspondant. Cette idée semble a priori fort logique.
Ce choix est celui actuellement retenu pour le filtre.

- 2) La catégorie cat2 doit être incluse dans les solutions concurrentes de cat1 (concurrentes au sens des intervalles de confiance). Cette idée aussi logique que la précédente est bien plus restrictive, trop restrictive même si on tient compte de l'aspect douteux des matrices de transition.

- 3) La catégorie cat2 est choisie librement par la règle la plus probable même en dehors des propositions de l'analyseur morphologique. Cette idée peut sembler bizarre, si elle introduit des anomalies elle corrige pourtant un certain nombre d'erreurs.

> Exemple : considérons le début de phrase "La souris appel ..." dans lequel le verbe a été coupé. En principe, pour un mot coupé (type "spect" pour spectacle) l'analyseur indique la catégorie pasd (pas de résultat). Ici, il ne va rien en être car "appel" sera analysé en tant que substantif. On voit bien que cette fois, la solution exacte se trouve en dehors des catégories indiquées par l'analyseur.

6.5 TAUX D'ERREURS

Rappelons et complétons, les éléments déjà listés dans le chapitre sur le Maître d'école qui interviennent dans la stratégie de correction d'erreurs :

- Sur une forme textuelle isolée :
  7. Nature des fautes typographiques :
     Taux de substitution,
     Taux d'omission
     Taux d'ajout
     Taux de permutation
     Taux de fautes claviers (erreurs de frappe quand la substitution d'un caractère est faite par celui d'une touche adjacente).

  8. Localisation des fautes :
     Taux de fautes portant sur la racine.
     Taux de fautes portant sur les désinences.
     Taux de fautes sur les verbes.
     Taux de fautes sur les substantifs.
     Taux de fautes sur les adjectifs.
     Taux de fautes sur les articles.
     Taux de fautes d'origine phonétique

- Taux sur des fautes provenant d'interactions entre formes textuelles

9. Coupure de mots ou de phrases
   Taux de fautes de coupure de mots
   Taux de fautes de soudure de mots
   Taux de fautes de coupure de phrases
   Taux de fautes de soudure de phrases

10. Fautes d'accord
    Taux de fautes d'accord dans le groupe nominal.
    Taux de fautes d'accord dans le syntagme verbal.
    Taux de faute d'accord entre le groupe nominal et le groupe verbal.

- Divers
  11. Taux du nombre de fautes par mots (taux pour une faute, pour deux fautes, etc.)

  12. Calcul de la *distance moyenne* entre chaque forme textuelle fausse et le mot choisi. On pourrait imaginer de nuancer cette distance moyenne par type de mots (substantifs, verbes …) ou bien par type de fautes.

  13. Evaluation des règles du type : flemme du rédacteur. Sur un mot faux, taux du cas : nombre de cas où la forme fausse comportait une marque du pluriel (x, s …) et la forme textuelle finale retenue était bien au pluriel / nombre total de cas où la forme fausse comportait une marque du pluriel (x, s …).

6.6 CONCLUSION : IMPLÉMENTATION À DEUX NIVEAUX

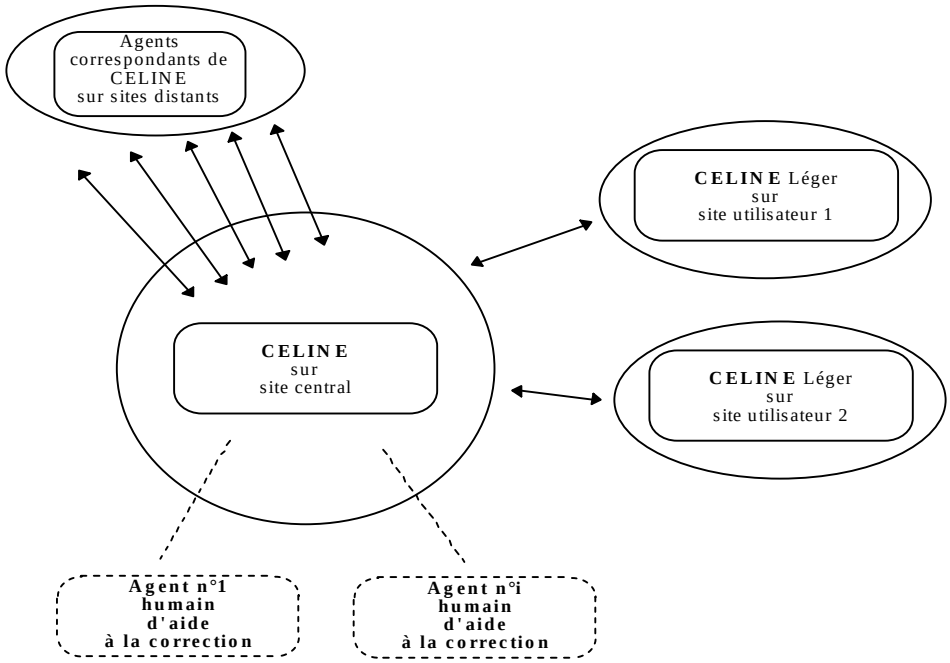


Figure 6.1 : implémentation à deux niveaux

Le système CELINE, se propose dans un premier temps d'être un prototype permettant d'expérimenter un certain nombre de concepts sur des corpus de taille non négligeable.

Nous visons une correction automatique. L'absence de sémantique rend cet objectif peu réaliste pour le moment. Toutefois, à terme, un système comme CELINE devrait permettre une correction des erreurs efficace dans un certain nombre de scénarios tel que celui d'une correction automatique lors des *reprises* pour un texte fréquemment complété et retravaillé. Pour pallier à l'absence de sémantique, la première correction et la correction de validation de la version finale se feraient en interaction avec l'utilisateur.

Si CELINE n'est pas destiné à concurrencer des produits commerciaux (correcteur de Word, Correcteur 101, Hugo, etc.), ce système pourrait être à la base d'un système industriel de traitement des erreurs sous la forme (figure 6.1) :

- Génération d'un ensemble léger implanté sur le site utilisateur et corrigeant à partir du MPLS les fautes courantes de l'utilisateur,
- Création d'un site central accessible par réseau avec des agents humains servant de correcteurs. L'intervention de ces agents humains pourrait être « en temps réel » ou en différé « question de coût d'abonnement ». Chaque demande d'intervention du site central correspond pour le site local à un apprentissage et fait évoluer le MPLS.

Pour comprendre les avantages d'un tel système, imaginons un scénario de correction lexicale et diverses stratégies :

- On commence par faire une recherche dans la table du MLPS.
- Si on ne trouve pas le mot, on lance localement un programme de traitement des mots inconnus, en utilisant les données de la table lexicale du MLPS et en s'aidant par exemple de la clé squelette.
- Si ce mode de correction échoue (c'est-à-dire, on ne trouve pas de forme textuelle remplaçante), le système local fait une recherche dans les lexiques de l'utilisateur sur son site (s'ils existent, par exemple PILAF) en utilisant des coefficients locaux.
- Si ce mode de correction échoue, le module local se connecte à CELINE qui va lancer une recherche multi-lexiques sur le réseau en utilisant ses connaissances (coefficient d'utilisation pour cet utilisateur) pour optimiser la recherche.
- Si cette recherche ne donne rien ou donne des solutions concurrentes non départagées par le système, alors on va faire intervenir l'agent humain. Mais quel agent ? Au lieu de faire appel au rédacteur, on pourrait imaginer qu'une société spécialisée dans la correction d'erreurs (une agence de presse ou de traduction automatique) offre à ses clients la possibilité de faire intervenir des "agents humains correcteurs" intégrés à CELINE qui bons linguistes corrigeraient sur leurs consoles les fautes non résolues par le système.

7. MÉTHODE DES STRUCTURES

Finalité
La notion de structure permet également de satisfaire certains besoins :

1. Le besoin d'une interface commune permettant, en vue de la correction des fautes d'accord, le traitement d'informations en provenance d'agents différents. Il fallait de toute façon une méta-représentation.
2. Le besoin d'une analyse linéaire émergeant par exemple de l'énoncé des règles d'accord tel qu'on les trouve dans les manuels (exemple : *lorsque le groupe nominal sujet comporte un adverbe de quantité comme beaucoup de, peu de, combien de, que de, etc. le verbe se met au pluriel*) ou bien des productions de certains automates d'états finis.
3. Le besoin de travailler sur des arbres ou des sous-arbres (productions les plus courantes des analyseurs syntaxiques).

Résumé
A partir des généralités ainsi dégagées, un paragraphe va tout d'abord présenter la notion de *structure* et de *qualité d'une structure*. Viendra ensuite un panorama des règles d'accord à implanter, dans lequel nous distinguerons des règles qualifiées de *règles d'unification* et des règles qualifiées de *règles d'anti-unification*. Cette distinction nous fera distinguer des *structures unifiables* basées sur des règles d'unification et des *structures absorbantes* basées sur des règles d'anti-unification. Pour terminer, nous examinerons une quantification du calcul de la qualité pour des structures terminales et des structures non terminales.

7.1 APPROCHE DE LA NOTION DE STRUCTURE

L'analyse syntaxique d'une phrase traduit les relations pouvant exister entre les mots composant la phrase, relations pouvant se représenter sous la forme d'un arbre. Parmi les règles grammaticales de la langue française, si quelques règles ont une portée générale, la majorité portent sur des sous-parties de la phrase (par exemple groupe nominal, groupe verbal etc.). Ces sous-parties de la phrase se traduisent, dans la représentation arborescente, sous la forme de sous-arbres que nous appellerons structures (structure du groupe nominal, structure du groupe verbal...).

Pour illustrer les rapports entre règles grammaticales et structures, nous allons prendre l'exemple d'une phrase entachée d'erreurs :

Les petite fille blonde et charmants étaient bronzées et parfumées

Nous remarquons que cette phrase passe sans problème l'étape de l'analyse lexico-morphologique puisque chacun des mots correspond à une forme acceptable. Nous donnerons le résultat de l'analyse syntaxique en utilisant une grammaire de dépendances. En ce qui concerne les propositions de correction nous évoquerons quelques heuristiques courantes.

7.1.1 FRAGMENTATION DE L'ARBRE SYNTAXIQUE

Si nous regardons les règles grammaticales possibles, concernant les accords applicables sur cette phrase, nous obtenons :

- Accord en genre et en nombre dans le groupe nominal entre le nom et ses adjectifs

- Accord en personne et nombre entre le groupe nominal sujet et le verbe
- Accord en genre et en nombre entre le ou les adjectifs qualificatifs attributs et le groupe nominal sujet.

Ces règles grammaticales nous suggèrent l'ensemble de la méthode à appliquer :

1. Fragmentation de l'arbre complet (figure 7.1) en trois sous-arbres :
   - le premier correspondant au groupe nominal sujet (figure 7.2),
   - le deuxième correspondant au verbe,
   - le troisième correspondant au groupe adjectif qualificatif attribut.

2. Réunion de ces structures élémentaires dans un meta-arbre[5] :
   - un groupe verbal (figure 7.3) réunissant les deux structures verbe et adjectifs qualificatifs attributs,
   - un groupe sous-phrase regroupant groupe nominal et verbe ou bien groupe nominal et groupe verbal.

3. Examen pour chaque structure de l'homogénéité des accords en fonction des règles grammaticales.

4. Au niveau du meta-arbre, examen pour chaque assemblage possible de l'unification éventuelle des accords. Nous appellerons *qualité* d'une structure l'ensemble unifié des variables morphologiques (genre, nombre, personne ...) définissant les rapports de la structure par rapport à celles qui lui sont rattachées.

5. En cas d'anomalie dans une structure ou au niveau des méta-constructions, application de règles heuristiques de choix. Le développement du choix d'une heuristique particulière sera développé au paragraphe 2.

L'arbre syntaxique, correspondant à la phrase exemple, peut être fragmenté en trois sous structures.

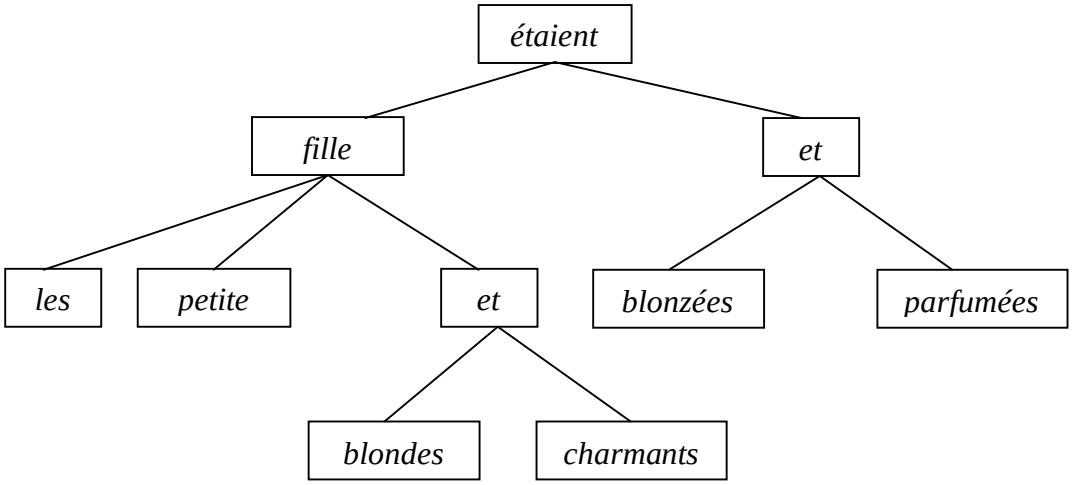


Figure 7.1 : arbre complet

[5] Nous utiliserons le terme de *meta-arbre* pour désigner un arbre complet ou un sous-arbre syntaxique, non développé, traduisant des liens entre des structures, par exemple (P(GN GV).

7.1.2 TRAITEMENT DU GROUPE NOMINAL

Les petite fille blonde et charmants

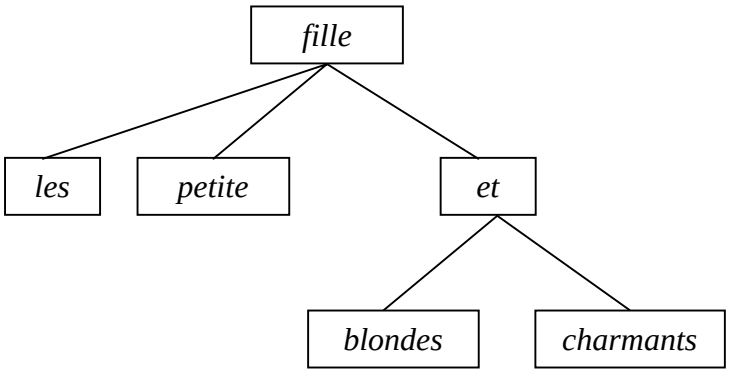


Figure 7.2 : groupe nominal

- **Examen du désaccord en genre :**

La règle heuristique du nombre minimal de correction (chapitre 2) nous propose par "trois voix contre une" le féminin.

La règle du maintien de la phonétique (chapitre II) nous conduit également par trois voix contre une à adopter le féminin (*petite/petit; fille/garçon; blonde/blond; charmantes/charmants*). On pourrait d'ailleurs considérer que le seul facteur de la distance phonétique entre *fille* et *garçon* impose le féminin.

La conjonction des deux règles semblant suffisante, nous pouvons donc attribuer la "qualité" morphologique *féminin* au groupe nominal. Dans la structure de données, les variables morphologiques correspondantes sont mises à féminin avec mémorisation d'un éventuel changement en vue d'une génération lexicale ultérieure.

- **Examen de l'accord en nombre :**

Cette fois les règles heuristiques précédentes ne permettent pas de conclure aussi facilement.

- La règle heuristique du nombre minimal de corrections nous propose par "trois voix contre deux" le singulier.
- La règle du maintien de la phonétique nous conduit par contre à adopter le pluriel.
- La règle heuristique du non-ajout d'information nous conduit au pluriel.

Nous pouvons admettre une indécision provisoire entre singulier et pluriel. Dans la structure de données nous reportons cette indécision sous la forme de la double présence des qualificatifs *sin* et *plu*.

7.1.3 GROUPE VERBAL

étaient bronzées et parfumées

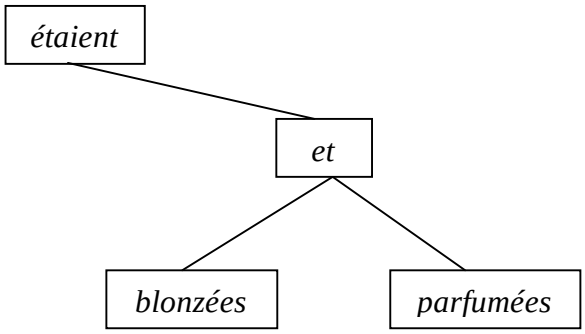


Figure 7.3 ; groupe verbal

La vérification des accords se fait en trois étapes en considérant la structure groupe adjectifs attributs, puis la structure verbe et pour terminer le groupe verbal union des deux.

Cette fois tout va bien, l'ensemble du groupement possède les qualités pluriel et féminin.

7.1.4 UNIFICATION

A l'étape ci-dessus l'unification du groupe verbal (féminin, pluriel) ne posait pas de problème puisqu'il était composé de l'union de deux structures ayant chacune la qualité féminin pluriel.

Par contre l'unification au niveau du groupe nominal et du groupe verbal pose problème au niveau de la variable nombre.

Nous pouvons considérer provisoirement, pour se fixer les idées, les deux règles heuristiques suivantes :

> 1. Toute structure ou meta-arbre comportant x mots (significatifs pour le nombre et/ou le genre) et écrite initialement sans anomalie d'accord, peut être considérée comme un îlot de confiance pour la détermination du nombre et/ou du genre.
> La valeur de x, initialement fixée à une valeur élevée (4 ou 5), pourrait évoluer dynamiquement par un suivi statistique.
>
> 2. En cas de doute sur la qualité d'une structure, la présence d'un îlot de confiance rattaché, permet de choisir la ou les variable(s) morphologique(s) douteuse(s) par adoption de celle(s) de l'îlot de confiance.

L'application de ces deux règles heuristiques nous permet de forcer, pour la qualité de la structure *groupe nominal*, le nombre à pluriel.

7.1.5 GÉNÉRATION DE LA PHRASE CORRECTE

Pour tous les mots dont une des variables morphologiques finales n'est pas incluse dans celles attribuées initialement par l'analyse morphologique, il y a lieu de procéder à une génération de la forme correcte.

Dans le cas de notre exemple, il faudra donc générer :
le pluriel de *petite*,
le pluriel de *fille*,

le pluriel de *blonde*,
le féminin de *charmants*,
avant d'obtenir la forme corrigée de la phrase :

Les petites filles blondes et charmantes étaient bronzées et parfumées

7.2 LES RÈGLES D'ACCORD

Nous avons déjà rencontré au chapitre 2, des règles grammaticales se mettant sous la forme :

(Observation d'un fait au niveau 1) ⇒ (Conséquence au niveau 2)

> Par exemple :
> *le Si n'est jamais suivi du futur ou du conditionnel , le présent en tient lieu*. Nous voyons que :
> Si + futur ou conditionnel ⇒ Verbe au présent.

Nous allons étoffer notre ensemble de règles applicables avant d'aborder les aspects quantitatifs..

7.2.1 ENONCÉS DES RÈGLES D'ACCORD

Ce paragraphe a pour objet d'expliciter un certain nombre de règles d'accord. Bien qu'il s'agisse là de données grammaticales sous une forme purement linguistique, leur lecture permet de mieux comprendre leur mise en place informatique dans l'agent DCFA de détection et correction des fautes d'accord.
Ces règles sont extraites du volume 3 de "Le nouveau Bescherelle". (Bescherelle 84) ou du Grevisse (Grevisse 69).

7.2.1.1 Accord du verbe avec le groupe nominal sujet

7.2.1.1.1 Il n'y a qu'un groupe nominal sujet :

A1 Le verbe se met à la même personne que le groupe nominal sujet
Remarque : Les règles A1 et A2 ci dessous peuvent se regrouper en un seul énoncé (comme par exemple les règles du § 7.2.1.2.1 sur les adjectifs ou du § 7.2.1.3.1 sur les participes passés). Nous les avons toutefois séparé en deux règles comme cela a été fait dans l'implémentation et le traitement quantitatif. Par le suite, par contre, nous avons maintenue des énoncés uniques.

Exemple : Nous devons vraiment partir.

A2 Le verbe se met au même nombre que le groupe nominal sujet.

Exemple : La souris mange le fromage.
Les petits enfants dormaient.

A3 Lorsque le groupe nominal sujet comporte un adverbe de quantité comme beaucoup de, peu de, combien de, que de, *etc., le verbe se met au pluriel.*

Exemple : Beaucoup de ces enfants chantent dans une chorale.

Lorsque l'adverbe de quantité a un sens partitif, le verbe se met au singulier.

Exemple : Peu de neige est tombée cet hiver.

A4 Lorsque le groupe nominal sujet représente un ensemble d'éléments (sujet collectif), le verbe se met soit au singulier soit au pluriel.

Exemple : Une foule de visiteurs se précipita.
Une foule de visiteurs se précipitèrent.

7.2.1.1.2 Il y a plusieurs groupes nominaux sujets :

B1 Lorsqu'il y a plusieurs groupes nominaux sujets, le verbe se met au pluriel.

Exemple : Jacques et Damien décidèrent d'aller au cinéma.

B2 Lorsque les groupes nominaux sujets sont de personnes différentes, le verbe se met à la personne de valeur numérique la moins élevée et au pluriel.

B2a - 2 ième personne + 3 ième personne : le verbe se met à la deuxième personne du pluriel.

Exemple : Marie et toi marcherez derrière.

B2b - 1er personne + 2 ième ou 3 ième personne : le verbe se met à la 1er personne du pluriel.

Exemple : Mes amis et moi voulions ce cadeau.

B3 Lorsque les deux groupes nominaux sujets sont réunis par comme, ou, ainsi que, avec, *le verbe se met soit au singulier soit au pluriel.*

Exemple : La bière comme le vin contient de l'alcool.
La bière comme le vin contiennent de l'alcool.

7.2.1.2 Accord de l'adjectif qualificatif

7.2.1.2.1 Accord de l'adjectif qualificatif épithète

A1 L'adjectif qualificatif épithète s'accorde en genre et en nombre avec le nom qu'il qualifie

Exemple : L'enfant ravi monta sur son vélo neuf.

A2 Lorsque l'adjectif qualificatif qualifie plusieurs noms, il se met au pluriel

Exemple : Les étrangers aiment la cuisine et la littérature .

A3 Lorsque l'adjectif qualifie plusieurs noms de genres différents, il se met au masculin plurie
l
Exemple : L'homme portait une chemise et un pantalon .

7.2.1.2.2 Accord de l'adjectif qualificatif attribut

B1 L'adjectif qualificatif attribut s'accorde en genre et en nombre avec le groupe nominal sujet

Exemple : Les enfants étaient heureux.

B2 S'il y a deux groupes nominaux sujets, l'adjectif attribut se met au pluriel : si ces deux groupes nominaux sont de genre différent, l'adjectif attribut se met au masculin pluriel

Exemple : La soupe et le poisson sont froids.

B3 Lorsque l'adjectif attribut est construit avec avoir l'air, *on peut soit le mettre au masculin singulier en l'accordant avec le nom air, soit l'accorder avec le sujet.*

Exemple : Elle a l'air bien sérieux.
Elle a l'air bien sérieuse.

7.2.1.3 Accord du participe passé

7.2.1.3.1 Participe passé employé comme épithète

A Le participe passé en fonction d'épithète s'accorde en genre et en nombre avec le nom qu'il qualifie

Exemple : Un homme averti en vaut deux.

7.2.1.3.2 Participe passé employé avec l'auxiliaire du verbe avoir

B1 Lorsqu'il n'y a pas de complément d'objet direct, le participe passé reste invariable

Exemple : Ils avaient couru comme des fous.

B2 Lorsque le complément d'objet direct se trouve après le verbe, le participe passé reste invariable

Exemple : Les enfants ont dévoré tous les gâteaux.

B3 Lorsque le complément d'objet direct se trouve placé avant le verbe, le participe passé s'accorde en genre et en nombre avec lui

Exemple : Tu n'as même pas regardé les fleurs que je t'ai offertes.

7.2.1.3.3 Participe passé employé avec l'auxiliaire être

C1 Le participe passé s'accorde en genre et en nombre avec le groupe nominal sujet

Exemple : Les feuilles des arbres étaient tombées.

C2 Le participe passé d'un verbe pronominal présente plusieurs cas d'accord :

*C2a Lorsque le pronom (*me, te, se, *...) est le complément d'objet direct du verbe (*se rencontrer, se baigner, se vendre, se sauver, *...), le participe passé s'accorde en genre et en nombre avec le sujet.*

Exemple : Elles se sont baignées dans la rivière.

*C2b Lorsque le pronom est le complément d'objet indirect du verbe (*s'acheter, se faire mal, se dire, etc*), le participe passé ne s'accorde ni en nombre ni en genre avec le sujet*

Exemple : Elle s'est dit qu'il ne viendrait pas.
Elles se sont lavé les mains.

C2c Le participe passé s'accordera avec le complément d'objet direct s'il est placé avant le verbe.

Exemple : Tu ne peux imaginer les choses que je me suis dites

7.2.1.3.4 Participe passé suivi d'un infinitif

Le participe passé conjugué avec avoir *et suivi d'un infinitif s'accorde avec le complément d'objet direct lorsque celui-ci traduit une action de ce complément.*

Exemple : les marins que j'ai vus *(en train de)* pêcher.
Les poissons que j'ai vu pêcher.

7.2.2 UNIFICATION DE STRUCTURES

Comme nous l'avons vu au paragraphe 7.1.2.4, l'unification d'une structure va consister à choisir pour chacun des éléments de la structure le même groupement des variables morphologiques. Ce groupement de variables unifiées est défini comme la qualité de la structure. Cette unification est rendue nécessaire dans deux cas :

1. L'analyse morphologique d'une phrase va proposer pour certains mots une alternative de variables morphologiques antagonistes. Nous qualifierons ce phénomène par le terme : "ambiguïté de variables morphologiques".

   Exemple : l'analyse morphologique de la structure correspondant au groupe nominal *le méchant chien jaune* va conduire à une ambiguïté de variables morphologiques pour *jaune* :

le	det	mas	sin	
méchant		adj	mas	sin
chien		subc	mas	sin
jaune		adj	mas	fem	sin

   Résoudre ce type d'ambiguïté demande de faire un choix à l'aide d'une heuristique :
   - En adoptant un choix statistique sur l'ensemble de la structure groupe nominal, nous pouvons affecter à *jaune* le genre *mas* et supprimer l'alternative *fem*.
   - En utilisant les grammaires de dépendance on peut aussi imposer le genre du substantif gouverneur.

2. Le texte initial peut contenir des fautes d'orthographes.

   Exemple : le groupe nominal *"le méchant chiens jaune"* va conduire cette fois à :

le	det	mas	sin	
méchant	adj	mas	sin	
chiens	subc	mas	plu	
jaune	adj	mas	fem	sin

> Nous avons en plus une faute d'accord en nombre. Si nous admettons que la correction la plus probable est celle d'une erreur sur *chiens,* nous devons :
> - affecter à *jaune* le genre *mas* et supprimer l'alternative *fem*
> - remplacer le pluriel *plu* de *chiens* par *sin*
>
> Après ces opérations les quatre mots de la structure sont (*mas, sin).* La structure a été unifiée et elle peut porter globalement la qualité (*mas ,sin).*

7.2.3 STRUCTURES UNIFIABLES ET RÈGLES D'UNIFICATION

Soit la règle GN = det adj subc adj
Si on considère la chaîne morphologique *det adj subc adj*, les variables morphologiques associées doivent posséder le même genre et le même nombre. Nous conviendrons de dire qu'il s'agit d'une *structure unifiable* et que la règle d'accord correspondante est une règle d'unification.

La qualité de la structure globale va être affectée par l'utilisation de règles d'origine heuristique, par exemple : calcul de poids affectés aux différentes variables, poids définis statistiquement. Nous détaillerons ce procédé au paragraphe 7.3.4.

7.2.4 STRUCTURES ABSORBANTES ET RÈGLES D'ANTI-UNIFICATION

Soit la règle GN = gn coco gn avec coco = {et}.

> Supposons que gn1 est du genre masculin et que gn2 est du genre féminin. Si nous considérons la chaîne morphologique *gn1 et gn2,* l'unification n'est pas possible puisque, la règle applicable d'accord en genre précise que si l'un des gn est masculin et l'autre féminin, le genre de GN est masculin.
>
> En ce qui concerne le nombre de la structure globale, la règle précise que le nombre de GN est considéré comme étant un pluriel.

Nous dirons qu'il s'agit d'une *règle d'anti-unification* : la reconnaissance d'un genre masculin pour une des deux structures gn ("*structure absorbante*"), va affecter à la structure globale le genre masculin. L'autre structure ne sera pas touchée par cette affectation de la qualité globale.

La qualité de la structure globale va pouvoir être affectée par l'application de règles grammaticales.

En sens inverse, l'affectation "extérieure non ambiguë" par exemple par un autre agent (agent humain) d'une qualité à une structure correspondant au cas d'application d'une règle d'anti-unification peut rendre impossible la répercussion de cette affectation aux sous-structures composantes.

Par exemple dans la phrase :

les parents et leur enfants jouent vifs et gais

rien ne peut permettre de décider s'il s'agit de *leurs enfants* ou de *leur enfant.* Tout au plus, on pourra penser qu'en appliquant la règle de la flemme du rédacteur, on peut admettre que *leurs enfants* est plus probable que *leur enfant.*

7.3 DÉFINITION QUANTITATIVE DE LA QUALITÉ D'UNE STRUCTURE

7.3.1 EXEMPLE SUPPPORT POUR LA DÉFINITION

Pour illustrer l'aspect quantitatif du calcul des qualités nous considérerons une phrase sans erreur[6] :

le beau racé chien jaune et les chattes courent furieux et affamés

Pour l'étude syntaxique, nous utiliserons la grammaire :

p	—>	gn	gv			
gn	—>	gn	coco(*et*)	gn		
gn	—>	det	adjq*	subc	adjq*	
gv	—>	verb	adja			
adja	—>	adjq*	ppas*	coco(*et*)	adj*	ppas*

L'analyse morphologique de la phrase donne après désambiguïsation :

le	det	sin mas tre cod
beau	adjq	sin mas
racé	adjq	sin mas
chien	subc	sin mas
jaune	adjq	sin mas fem
et	coco	
les	det	plu fem mas tre cod
chattes	subc	plu fem
courent	verb	plu tre pre sub pre ind
furieux	adjq	sin plu mas
et	coco	
affamés	ppas	plu mas

Remarque sur les renseignements amenés par l'analyse morphologique :
Lorsqu'on s'intéresse aux valeurs d'une variable morphologique, nous nous apercevons de la nécessité de faire plus qu'un simple comptage de chacune des marques morphologiques puisque, par exemple, si nous nous intéressons au genre nous ne le trouvons exprimé que 7 fois sur 12 mots soit à peine plus de 50 %. Pour tirer des conclusions statistiques pertinentes sur une phrase nous allons être conduits à gérer le nombre de mots porteurs de renseignements sur la variable morphologique ciblée.

L'arbre syntaxique décoré donné sous la forme de l'ordre lexicographique[7] des chemins va être :

1.
1.1.gn
1.1.1.gn

---

[6] Phrase possédant la même structure que celle du chapitre d'introduction (*le beaux cheval …*)

[7] L'ordre lexicographique des chemins est utilisé par un des agents pour mémoriser les relations entre structures (vois annexe D).

1.1.1.1.det(le,mas,sin,tre,cod)
1.1.1.2.adjq(beau,mas,sin)
1.1.1.3.adjq(racé,mas,sin)
1.1.1.4.subc(chien,mas,sin)
1.1.1.5.adjq(jaune,mas,fem,sin)
1.1.2.coco(et)
1.1.3.gn
1.1.3.1.det(les)
1.1.3.2.subc(chattes)
1.2.gv
1.2.1.verb(courent)
1.2.2.adja
1.2.2.1.adjq(furieux)
1.2.2.2.coco(et)
1.2.2.3.ppas(affamés)

7.3.2 VALEURS INITIALES D'UNE STRUCTURE TERMINALE

Pour permettre un calcul, les variables morphologiques de type chaînes de caractères ('*sin*', '*mas*', ...) vont être remplacées par des nombres :

Variable :	*mas*	*fem*	*sin*	*plu*
Valeur :	+1	-1.	-1	+1

Ces valeurs sont choisies de manière à ce que la contribution d'une alternative (mas/fem ou sin/plu) soit nulle.

Par ailleurs, pour chaque variable morphologique possible, certains mots sont porteurs de renseignements et d'autres non. Par exemple, dans le groupe *"furieux et affamés"* un seul (*affamés*) est significatif pour le nombre.
En vue d'un traitement quantitatif, il y a lieu de faire intervenir ce fait sous peine de voir disparaître la signification des nombres obtenus ou de voir apparaître un doute sur les accords dans le cas d'une phrase pourtant sans erreur. On ne peut pas se contenter de compter simplement les marques de chaque type (voir exemple ci-dessous).

Une solution efficace consiste à compter les mots significatifs par rapport à la variable morphologique considérée. Pour cela, on va associer à toute structure terminale un ensemble de 5 nombres :

- Un nombre "*genre*" g, entier relatif tel que :
  g = 0 si les variables morphologiques données par l'analyse morphologique
  ne contiennent pas *mas* ou *fem*,
  ou bien
  contiennent les deux à la fois.
  g = 1 si on trouve seulement *mas*,
  g = -1 si on trouve seulement *fem*.

- Un nombre *"nombre"* n, entier relatif tel que :

n = 0 si les variables morphologiques données par l'analyse morphologique
ne contiennent pas *sin* ou *plu*,
ou bien
contiennent les deux à la fois.
n = 1 si on trouve seulement *plu*,
n = -1 si on trouve seulement *sin*.

- Le "*nombre total de mots*" nbmt de la structure :
  nbmt = 1 pour une structure terminale.

- Le "*nombre de mots porteurs de renseignement sur le genre*" nbmg :
  nbmg = 0 si g = 0
  nbmg = 1 si g ≠ 0.

- le "*nombre de mot nbmn porteurs de renseignement sur le nombre* " nbmn :
  nbmn = 0 si n = 0
  nbmn = 1 si n ≠ 0.

La qualité de la structure va être représentée, dans une notation condensée, par les cinq nombres précédents séparé par des slashs "/" :

genre/nombre/nbmt/nbmg/nbmn/

Dans le cas de notre exemple nous obtenons alors:

Mots	Analyse morphologique	genre	nombre	nb mot total	nb mot genre	nb mot nombre	Qualité associée
le	mas, sin	+1	-1	1	1	1	1/-1/1/1/1/
beau	mas, sin	+1	-1	1	1	1	1/-1/1/1/1/
racé	mas, sin	+1	-1	1	1	1	1/-1/1/1/1/
chien	mas, sin	+1	-1	1	1	1	1/-1/1/1/1/
jaune	mas, fem, sin	0	-1	1	1	1	0/-1/1/0/1
et		0	0	1	0	0	0/0/1/0/0/
les	mas, fem, plu	0	+1	1	0	1	0/1/1/0/1/
chattes	fem, plu	-1	+1	1	1	1	-1/1/1/1/1/
courent	plu	0	+1	1	0	1	0/1/1/0/1/
furieux	mas, sin, plu	+1	0	1	1	0	1/0/1/1/0/
et		0	0	1	0	0	0/0/1/0/0/
affamés	mas, plu	+1	+1	1	1	1	1/1/1/1/1/
Totaux *:*				12	8	9	

7.3.3 QUALITÉ D'UNE STRUCTURE NON TERMINALE

Comme pour toute structure terminale, la qualité d'une structure non terminale va être définie par un ensemble de cinq nombres calculés à partir des qualités des structures composantes :

le *genre* g tel que -1<=g<=1,

le *nombre* n tel que -1<=n<=+1,

le *nombre total de mots* de la structure : nbmt,

le *nombre total de mots porteurs de renseignement sur le genre* : nbmg.

le *nombre total de mots porteurs de renseignement sur le nombre* : nbmn.

Soit la structure comportant n sous-structures dont les qualités respectives sont :

$genre_1/nombre_1/nbmt_1/nbmg_1/nbmn_1/(\ .)$
...
$genre_i/nombre_i/nbmt_i/nbmg_i/nbmn_i(\ .)$
...
$genre_n/nombre_n/nbmt_n/nbmg_n/nbmn_n(\ .)$

Nous admettrons que les *genre* et *nombre* de la structure unifiée sont les moyennes pondérées des nombres et genres de chacune des sous-structures. Les coefficients de pondération sont le produit du nombre de mots porteurs de renseignement correspondant par un coefficient a_{ik} provenant de la règle utilisée et que nous découvrirons au paragraphe VII.3.4. Dans l'exemple nous supposerons provisoirement que tous les a_{ik} sont égaux à 1 ce qui correspond aux valeurs initiales de ces coefficients si nous optons pour une évaluation statistique.

$$genre := \frac{\sum_{i:=1}^{n} (genre_i . a_{ig} . nbmg_i)}{\sum_{i:=1}^{n} (a_{ig} . nbmg_i)} \qquad nombre := \frac{\sum_{i:=1}^{n} (nombre_i . a_{in} . nbmn_i)}{\sum_{i:=1}^{n} (a_{in} . nbmn_i)}$$

Les nombres de mots sont obtenus par sommation :

$$nbmt = \sum_{i:=1}^{n} nbmt_i \qquad nbmg = \sum_{i:=1}^{n} nbmg_i \qquad nbmn = \sum_{i:=1}^{n} nbmn_i$$

Exemple détaillé : unification de la structure du groupe adjectif attribut

1/0/1/1/0/adj(furieux,mas,sin,plu) 0/0/1/0/0/conj(et) 1/1/1/1/1/adj(affamés,mas,plu)

Structure à unifier	Genre	Nombre	nbmt	nbmg	nbmn
1/0/1/1/0/adj(furieux,mas,sin,plu)	1	0	1	1	0
0/0/1/0/0/conj(et)	0	0	1	0	0
1/1/1/1/1/adj(affamés,mas,plu)	1	1	1	1	1
Total			3	2	1
Qualité	1	1	3	2	1

Nous obtenons alors comme structure unifiée du groupe adjectif attribut :

1/1/3/2/1/adja(furieux et affamés, mas, plu)

Autres exemples sans détail des calculs :

Structure à unifier	Genre	Nombre	nbmt	nbmg	nbmn
0/1/1/0/1det(les,mas,fem,plu) -1/1/1/1/1/subc(chattes,fem,plu)	-1	1	2	2	2
1/-1/1/1/1/det(le,mas,sin) 1/-1/1/1/1/adjq(beau,mas,sin) 1/-1/1/1/1/adjq(racé,mas,sin) 1/-1/1/1/1/subc(chien,mas,sin) 1/-1/1/1/1/adjq(jaune,mas,fem,sin)	1	-1	5	4	4
0/1/1/0/1/verb(courent,plu) 1/1/3/2/1/adja(furieux et affamés,sin,plu)	1	1	4	2	2

7.3.4 POIDS DANS LES RÈGLES D'UNIFICATION ET D'ANTI-UNIFICATION

Nous allons encore "affiner" ce calcul en considérant le type particulier de la structure traitée.

7.3.4.1 Coefficients de pondération du genre et du nombre dans une règle

7.3.4.1.1 Coefficients de pondération

Dans une règle du type GN = det adj subc adj, on peut s'interroger sur l'équivalence de la validité des informations portées par chaque constituant. Nous allons affecter à chaque constituant un doublet de coefficients traduisant la pertinence de ce constituant l'un par rapport au genre et l'autre par rapport au nombre (*coefficient de pertinence du nombre* cpn et *coefficient de pertinence du genre* cpg)

L'interprétation de cette pertinence peut être diverse. Nous allons voir par exemple une interprétation "logique" et une interprétation "statistique".

7.3.4.1.2 Interprétation logique

Les substantifs possèdent pour la plus part un genre unique ce qui n'est pas le cas pour les adjectifs. Nous pouvons en déduire qu'un adjectif est plus significatif pour le genre et lui affecter un fort coefficient pour le genre (par exemple 3).
En sens inverse pour le nombre, par rapport aux adjectifs, plus de substantifs ont une graphie différente entre le singulier et le pluriel et de ce fait sont plus expressifs pour le nombre.

Les deux hypothèses ci-dessus sont discutables mais une exploitation systématique du dictionnaire permettrait de calculer rigoureusement de tels coefficients pour chaque catégorie. Nous pouvons aussi considérer que ces coefficients traduisent quantitativement des règles heuristiques comme par exemple la *règle priorité au gouverneur* dans les structures de dépendances (Genthial 91).

7.3.4.1.3 Interprétation statistique

On peut aussi considérer que les coefficients traduisent un facteur de vérité rattaché à la non-probabilité d'erreurs. Par exemple on peut admettre qu'on peut oublier un "*e*" à un adjectif mais que, par contre, on ne commet pas d'erreurs entre un "*le*" et un "*la*". Cela conduit à affecter un fort coefficient "genre" au déterminant.

On peut même aller plus loin et admettre que les déterminants étant des mots de manipulation simple[8], on peut aussi affecter une valeur de vérité importante au nombre.

Une enquête statistique permettrait d'obtenir des valeurs initiales pour ces coefficients. Ce choix arbitraire initial pourrait ensuite évoluer toujours statistiquement par un suivi des fautes commises par un utilisateur donné. Cette méthode permettrait d'ajuster statistiquement les coefficients pour chaque utilisateur et de suivre son évolution (Menézo 93).

En s'inspirant des coefficients proposés par Genthial (Genthial 94) nous pourrions alors choisir pour les coefficients de pertinence des expressions récursives du type :

$$K = K \times \frac{1+nbfe}{1+nbfc}$$

avec nbfe : nombre de formes exactes et nbfc nombre de formes corrigées

Une telle formulation présente deux avantages :

- Elle ne demande pas forcément de valeurs initiales (suppression d'un délicat travail de statistique préalable). Les valeurs initiales des K pourraient être toutes mises à 1 ce qui revient à annuler l'effet de ces coefficients.
- La mémorisation de données est d'un coût minimum puisque seules les valeurs évolutives des coefficients sont en permanence conservées. Lors de la mise en œuvre du correcteur, il y a totalisation pour chaque règle employée et pour chaque catégorie lexicale concernée des *nbfe* et *nbfc*. Les coefficients de pertinence sont mis à jour à la fin de l'utilisation du correcteur ou à la fermeture de l'application si on imagine un correcteur interactif permanent "temps réel",

7.3.4.2 Formalisme d'écriture des coefficients de pertinence

Admettons des chiffres à titre d'exemple : det(1,1), adj(3,1), subc(1,3), le premier des coefficients s'adressant au genre et le deuxième au nombre.
Nous écrirons alors la règle GN —>1/1/det( )3/1/adj( )1/3/subc( )3/1/adj( ).

7.3.4.3 Exemple de calcul avec coefficients de pondération

Mots	Cat	coeff genre	Coeff nombre		Genre	Nombre
le	det	1	1	mas,sin	1*(+1)	1*(-1)
méchant	adj	3	1	mas,sin	3*(+1)	1*(-1)
chiens	subc	1	3	mas,plu	1*(+1)	3*(+1)
jaune	adj	3	1	mas,fem,sin	3(+1-1)	1*(-1)
					5/8	0

Dans ce cas, le jeu arbitraire de coefficients choisis ne permet pas de conclure sur le nombre à affecter à la structure.

7.4 CONCLUSION DU CHAPITRE

Par une approche méthodologique ou linguistique, nous avons

1. défini la notion de structure,
2. défini qualitativement la qualité d'une structure,

[8] Comme la majorité des mots grammaticaux (Becherelle Tome 3 p 165)

3. introduit une quantification de l'évaluation de la qualité.

Par ailleurs, nous avons établi une liste de règles d'accord sous une forme verbale.

Tous les cas ou besoins linguistiques ne peuvent être résolus par cette méthode, mais si elle permet le traitement des règles d'accord données, nous pensons que dans l'hypothèse d'un système multi-agents, l'agent correspondant DCFA aura déjà une forte activité de coopération avec les autres agents.

8. UNE APPROCHE DE L'INTELLIGENCE ARTIFICIELLE DISTRIBUÉE ET DES SYSTEMES MULTI-AGENTS

Finalité

Le prochain chapitre va présenter CELINE sous l'angle d'un système multi-agents. Les chapitres précédents ont peu à peu établi un cahier des charges informel pour CELINE. Dans le but d'une adéquation entre le choix du multi-agents et ce cahier des charges, il y a donc lieu de se mettre dans l'esprit des systèmes multi-agents puis de vérifier si les exigences du cahier des charges correspondent bien aux définitions en vigueur dans les systèmes multi-agents.

L'objet de ce chapitre est de présenter un résumé synthétique sur l'IAD (Intelligence Artificielle Distribuée), les systèmes multi-agents et la notion de tableau noir en se plaçant au niveau, non pas d'un expert de l'IAD réalisant un état de l'art complet, mais de celui d'un chercheur en langues naturelles désireux d'utiliser dans son domaine spécifique, le vocabulaire, les méthodes et les apports de l'IAD.

Pour un profane en la matière, la terminologie utilisée actuellement dans ce domaine semble relativement lourde car très hétérogène d'une application à une autre. Pour un domaine encore nouveau en attendant des définitions synthétiques, cette hétérogénéité est en réalité le signe d'une richesse et d'une adaptation des concepts généraux du domaine à la solution particulière utilisée, chaque solution particulière définissant ses propres termes en fonction du contexte.

Ce chapitre est présenté plutôt comme un résumé synthétique de différentes possibilités de choix en vue d'une mise en œuvre d'un système multi-agents. Cette approche m'a été facilitée par la lecture de deux thèses très synthétiques : celle de Michel Occello (Occello 93) en ce qui concerne les tableaux noirs et tout particulièrement les tableaux noirs parallèles et celle de Marie-Héléne Stéfanini (Stéfanini 93) en ce qui concerne les relations entre système multi-agents et linguistique.

Résumé

Une première partie va nous permettre de traiter successivement les points suivants :

- Comment l'utilisation conjointe du parallélisme et de l'intelligence artificielle permet de définir de nouveaux domaines et parmi ceux-ci le domaine des systèmes multi-agents.
- La notion d'agent et les regroupement d'agents : les sociétés. Nous dégagerons un ensemble de qualificatifs et de propriétés pouvant s'appliquer aux agents.
- Au sein d'une société l'une des préoccupations consiste à coordonner les activités des agents d'où le besoin de communiquer. Cette communication généralement définie comme le comportement social de l'agent, peut prendre des formes extrêmement variables depuis la simple réponse à un stimuli d'un agent réactif jusqu'à une communication faisant intervenir les notions les plus fines du génie cognitif et aboutissant à des négociations entre agents. La notion de contrôle explicite la coopération entre les agents.

Une deuxième partie décrira quelques systèmes complets en se limitant au domaine du traitement des langues naturelles.

La troisième partie va avoir pour objectif de valider CELINE comme un système multi-agents. Cette validation se poursuivra au chapitre suivant.

8.1 DE L'INTELLIGENCE ARTIFICIELLE AUX SYSTÈMES MULTI-AGENTS

La diversité des connaissances et des modes de raisonnement qui collaborent à la résolution d'un problème rendent difficile la réalisation d'un système monolithique capable de tout résoudre. L'IAD a pour but la résolution de problèmes d'IA par la distribution de l'intelligence en modules appelés *agents* qui *interagissent*. Le système unique va devoir être décomposé en sous-systèmes plus petits.

8.1.1 DESCRIPTEURS DES SYSTÈMES

Decker (Decker 87) classe les travaux de recherche et les systèmes multi-agents selon quatre critères :

- Granularité de la décomposition,
- Capacité de raisonnement
- Méthode de coordination,
- Mode de communication.

Demazeau (Demazeau 92) et le groupe PLEIAD[1] précisent ces quatre critères et utilisent :

- Le type d'agent défini par :
  - Le comportement de l'agent (autonomie)
  - L'homogénéité/hétérogénéité de la structure interne des agents,
  - La granularité, mesure du degré de complexité des agents,
- Le type d'implantation (mémoire partagée ou communication par message)
- Le type de contrôle mis en œuvre (centralisé, décentralisé, semi-centralisé)
- La densité, nombre plus ou moins élevé des agents.

Ces critères descripteurs des systèmes vont être définis progressivement au cours de ce chapitre.

8.1.2 GRANULARITÉ DE DÉCOMPOSITION

La décomposition d'un système unique (figure 8.1) peut se faire avec une granularité variable :

- Une *granularité fine* est caractérisée par une *décomposition au niveau instruction* et définit l'IA parallèle. L'IA parallèle se préoccupe des problèmes de performance des programmes et vise le développement d'algorithmes, langages et systèmes parallèles.
- Une granularité moyenne ou grosse est caractérisée par *une décomposition au niveau tâche* **;** elle définit la classe des *Systèmes Experts Distribués* SED.

1 PLEIAD : Pôle et Lieu d'Echanges en Intelligence Artificielle Distribuée

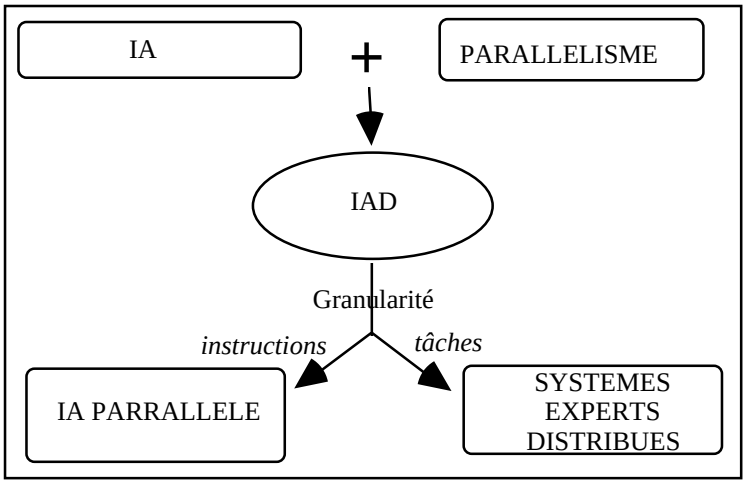


Figure 8.1 : IA parallèle et S.E.D.

IA parallèle et systèmes experts distribués diffèrent fortement par les méthodologies mises en œuvre et les solutions adoptées et constituent pour l'instant deux domaines distincts.

Pour la suite, nous allons considérer uniquement les systèmes experts distribués.

8.1.3 CAPACITÉ DE RAISONNEMENT

Selon leur capacité de raisonnement (figure 8.2), les Systèmes Expert Distribués (SED) peuvent être divisés en :

- *Systèmes Multi-Agents* SMA)
- *Résolveurs de Problèmes Distribués* (RDP).

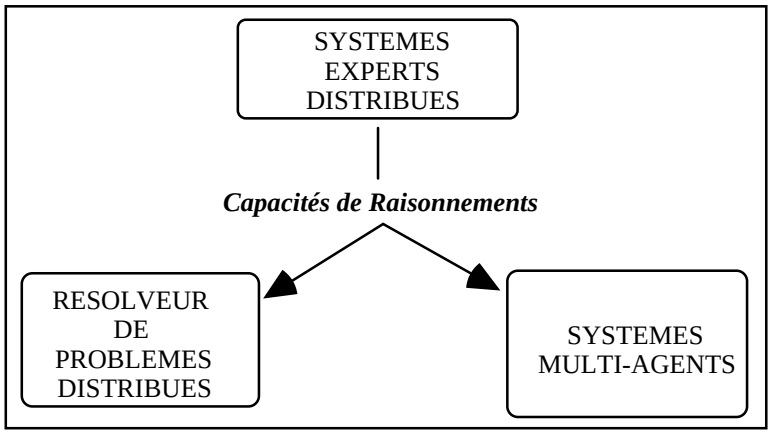


Figure 8.2 : S.M.A .et R.D.P.

Pour définir la répartition entre SMA et RDP nous allons donner quelques règles de classification.

Auteurs	Position sur SMA	Position sur RPD
Bond (88)	N'est pas dédié à un problème particulier. Un ensemble d'agents autonomes partagent connaissances et techniques mais n'ont pas d'organisation préalable. Les agents raisonnent sur leur organisation en fonction du problème à traiter. Le rôle des agents dans la résolution n'est pas toujours le même, l'agent possède des connaissances sur son comportement et la façon dont il peut prendre part à la résolution.	Tout système dédié à la résolution d'un problème particulier où différents modules coopèrent au niveau du partage de connaissances sur le problème et sur les techniques de résolutions.
Durfee (89)	Un agent de SMA, contrairement aux modules des RPD, n' a pas à connaître le but global pour agir.	
Decker (87)	Les SMA se basent sur des agents de moyenne granularité, plus spécialistes d'un sous problème et moins soucieux de l'aspect comportemental.	Tout système expert distribué, de forte granularité, composé d'agents complets, constituant de véritables systèmes experts.
Hynynen (89) Genesereth (86)	Pas de volontariat : les agents sont autonomes et réagissent seuls par rapport au contexte.	Agents "volontaires", bien connus, prévus pour coopérer : un seul concepteur est requis pour tous les agents, pour prévoir une entraide et proscrire les conflits dès la conception

Nous pouvons remarquer que l'approche RPD semble peu adaptée au traitement des langues naturelles :

- La constitution d'un RPD demande d'établir une décomposition a priori du système à étudier avec une coopération des agents de manière à atteindre un but commun à tous. Les systèmes TLN proposent en général des agents, au départ indépendants, spécialistes d'un domaine particulier, et possédant un but spécifique (analyseur ou générateur morphologique, analyseur syntaxique, etc.).
- Les stratégies à utiliser dépendent fortement de la phrase à analyser, établir a priori une planification ne semble pas présenter une "souplesse" suffisante.

Pour la suite, nous ne considérerons plus que les systèmes multi-agents.

"*Les systèmes multi-agents apportent bien plus que des découpages de connaissances : ils renouvellent notre façon d'aborder un problème et de concevoir ce qu'est le raisonnement et l'intelligence. Le problème ne se résout plus en partant d'un état initial pour arriver à un état final, mais en construisant au fur et à mesure des solutions partielles qui se trouvent sur le chemin, chaque agent cherchant à apporter des éléments de solution au fur et à mesure de l'exploration"*. (Ferber 89).

8.1.4 MÉTHODE DE COORDINATION ET MODE DE COMMUNICATION

Dans tous les cas les agents doivent coopérer et communiquer. Ces deux dimensions vont permettent une classification des systèmes multi-agents mais avant toute nouvelle classification nous devons approfondir nos connaissances sur les agents, les sociétés puis sur le contrôle de leurs interactions.

"*Les agents doivent prendre des décisions, pour leurs résolutions et leurs communications, basées sur des vues locales qui peuvent être incomplètes, incohérentes*

ou périmées. Ils doivent utiliser leurs possibilités de communication et ressources de calcul, non seulement pour la résolution du problème du domaine, mais aussi pour contrôler leurs actions et interactions" (Durfee, 89).

8.2 NOTIONS D'AGENTS

8.2.1 DÉFINITION

Le terme d'agent peut être interprété de diverses manières. Le groupe de travail de l'AFCET-IAD tend à définir une taxinomie du domaine et en donne la définition suivante :

Un *agent* est
- une *entité autonome*,
- qui poursuit un *objectif individuel*,
- qui est apte
  - à *agir sur l'environnement du système* auquel il appartient,
  - et/ou
  - à *interagir avec les autres agents*
- qui ne dispose que d'une *représentation évolutive* de cet environnement,
- qui peut *percevoir les autres agents* (communication, observation).

L'autonomie de l'agent (autonome, semi-autonome ou dépendant) est caractérisée par les capacités de flexibilité et d'adaptation de l'agent.

La *structure d'un agent* comprend :
- des *connaissances* (connues initialement ou acquises par l'agent)
- des *buts*
- des *actions* (sur l'environnement ou sur lui-même).

Les *mécanismes de traitement d'un agent* se divisent en :
- *mécanismes de raisonnement d'un agent* qui *construisent* les différentes actions possibles en fonction des buts et des connaissances de l'agent.
- *mécanismes de décision d'un agent* qui en fonction des buts et des hypothèses *choisissent* la ou les actions les plus appropriées.

Nous reprendrons au paragraphe 8.3.4 la structure d'un agent et les mécanismes de traitement.

8.2.2 LES SOCIÉTÉS ET SOUS-SOCIÉTÉS

Les agents peuvent être regroupés en *société* ou en *sous-société*. La constitution de groupes d'agents induit des problèmes liés à la coordination et à la coopération entre agents qui sont appelés *problèmes sociaux* entre agents.

La société se définit par
- les agents ou les sous-sociétés qui la composent,
- les liens qui les unissent et leurs interdépendances.

La *structure d'une société* comprend :

- un *langage d'interaction*,
- des *protocoles d'interaction.* Un protocole est un ensemble de règles, connues par tous les agents, traitant de l'enchaînement des interactions. Le protocole traduit l'aspect dynamique de la communication.

Les *mécanisme de traitement d'une société* se divisent en :

- *mécanismes de raisonnement d'une société* qui correspondent aux échanges possibles dans un protocole de résolution donné.
- *mécanismes de décision d'une société* qui utilisent des protocoles de résolution de conflit ou de négociation qui restructure l'organisation de la société.

L'*organisation d'une société* peut se faire selon deux modèles :

- un *modèle hiérarchique* de sociétés,
- un *modèle collégial* de sociétés.

8.2.3 CLASSIFICATION DES DIFFÉRENTS TYPES D'AGENTS

De nombreuses descriptions de la structure interne d'un agent sont données dans la littérature du domaine.

Par exemple, une description interne d'un agent a été élaborée par Jean Erceau et Jacques Ferber (Erceau et Ferber, 91) à partir de :

- l'aspect personnel regroupant les capacités de perception, de décision, d'action et de raisonnement sur les comportements de soi et des autres.
- l'aspect social qui concerne les capacités de communication et l'organisation de ces communications.

En IAD, on distingue deux écoles :

- l'École Cognitive

  Les *Systèmes Cognitifs* sont constitués d'un petit nombre d'agents de forte granularité, assimilables à des systèmes experts, une large place étant donnée aux aspects relationnels entre agents.

- l'École Réactive :

  Les *Systèmes Réactifs* visent un comportement intelligent par la coopération d'agents de faible intelligence réagissant simplement aux événements du monde. La granularité de ces agents est faible ; ils n'ont pas connaissance des buts globaux et n'ont que de très faibles capacités de raisonnement sur la coopération. Un comportement intelligent global résulte du travail de ces agents non-intelligents (*phénomène d'émergence*).

Agents Réactifs / Agents Cognitifs

Agents Réactifs (non délibératifs)	Souvent en grand nombre De plus bas niveau Souvent homogène Aucune mémoire locale Seule activité : effectuer une action donnée, à la détection dans l'environnement du stimulus attendu
Agents Cognitifs	Peu d'agents Forte granularité Souvent hétérogène Mode social d'organisation Possibilité d'avoir une mémoire locale Capacité de raisonnement et de décision explicites Représentation explicite de leur environnement Possibilité de connaissances sur les capacités des autres agents

A noter que cette classification ne fait pas l'unanimité et que certains travaux s'orientent vers des agents mixtes (Boissier 93).

Classification des agents cognitifs	
Agent Cognitif Communicant	Dispose d'outils de communication complets. Les informations qu'il échange sont liées au domaine de l'expertise. Aucun échange de vues sur des relations comportementales
Agent Cognitif Rationnel	Capacités de perception et de communication Capable de modifier son comportement de manière indépendante du domaine d'expertise. Capacités de raisonnement et d'apprentissage pour le comportement.
Agent cognitif spécialiste	Connaissances sur le comportement des autres agents, sur leurs capacités. Capacité de négociation : ententes avec d'autres pour mener une tâche.
Agent Cognitif Intentionnel	Toutes les capacités précédentes . Possède des buts, des plans locaux. Peut jouer des rôles différents en fonction de la situation. Maximum de place aux relations sociales (aspect comportemental : mécanismes d'intention, d'engagement)
Agent Cognitif social	Toutes les capacités précédentes. Possède les modèles explicites des autres agents. Raisonne et manipule des plans concernant les modèles des autres agents.

8.2.4 LE COMPORTEMENT D'UN AGENT

Un agent est caractérisé par différents types d'états mentaux : croyance, intentions préférences, etc. La représentation des états mentaux peut se faire par l'utilisation de prédicats, de graphes conceptuels, d'objets, de frames.

Les divers types de raisonnement d'un agent se ramènent à des comportements fondamentaux tels que :

- la prise de décision,
- la révision des croyances,
- la capacité à émettre des hypothèses.

Dans le détail, les principaux mécanismes de raisonnement d'un agent sont les suivants :

- le raisonnement hypothétique et la prise de décision,

- l'évaluation d'alternatives en fonction de ses préférences,
- le respect ou la violation de prescriptions (obligations, interdictions),
- le respect ou non des engagements de l'agent,
- le raisonnement sur autrui,
- la révision des croyances.

En fait, ces divers modes de raisonnement sont liés et doivent être mis en œuvre de façon concurrente par les agents. On peut représenter schématiquement ces interrelations dans un graphe (Boury-Brisset 94) (Rousseau 93):

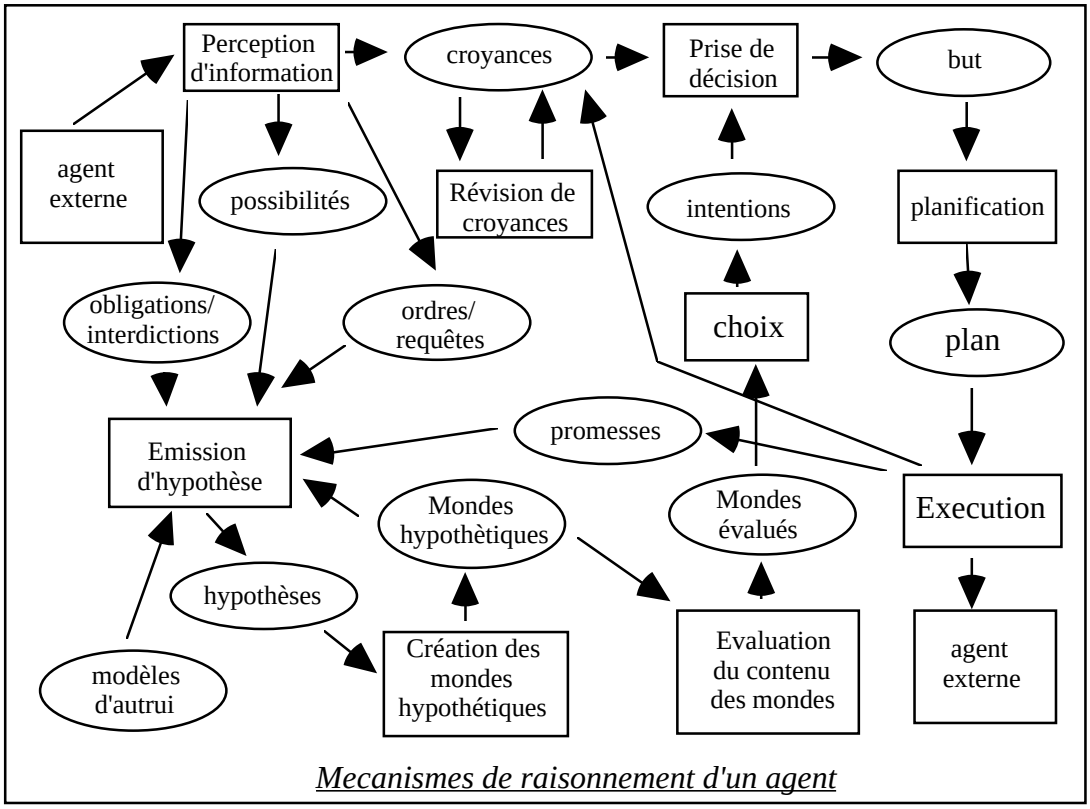


Figure 8.3 : Raisonnement d'un agent

8.2.5 COORDINATION

Les aptitudes des différents types d'agents à coopérer et à communiquer permettent d'envisager différentes organisations du contrôle de la coordination. Les méthodes de contrôle sont nombreuses et sont fonction des capacités des agents.

Quelques Exemples		
Coopération sans communication	Cas des agents réactifs	(Genesereth 86)
Organisation maîtres/esclaves	Communication extrêmement réduite	
Organisation statique	Lorsque l'on connaît bien le problème à traiter et qu'il se présente toujours sous des formes voisines. Un outil centralisé peut être seul à gérer l'activité des autres agents.	(Lesser and Corki 83) (Fox 81)
Planification mono-agents *Planification multi-agents*	Un agent (ou un groupe d'agents) détient (détiennent) un *plan de résolution.* Ce plan peut être centralisé dans un agent ou distribué dans plusieurs agents	(Georgeff 86) (Lanski 87) (Durffe and Lesser 87)
Organisation dynamique	Demande des agents évolués capables de *négocier*	Réseau de contrats (Smith et Davis 87)

8.2.6 EXEMPLES D'AGENTS PARTICULIERS

8.2.6.1 L'éco-résolution, exemple de système purement réactif

L'*éco-système* est une technique de résolution de problèmes fondée sur les principes des systèmes réactifs. Elle s'applique dans le cas où l'environnement évolue et modifie le fonctionnement global du système.
Comportement dit "t*ropique*" des agents avec des actions de base en termes de :

satisfaction locale,
rejet local,
réaction de survie,
satisfaction,
fuite,
agression,
etc.

8.2.6.2 Agents des architectures hybrides

Ils prennent des formes (et des noms !) variables en fonction du contexte et du type d'application ciblée.

8.2.6.3 Agents artificiels et agent humain

Dans un univers de prise de décisions stratégiques, l'agent humain coopère avec des agents artificiels pour la résolution de son problème. Il fixe l'objectif principal destiné aux agents artificiels et il évalue la solution que lui propose ces agents. Il décide du changement éventuel de l'objectif global ou de la création d'un autre plan d'actions dans le cas où des contradictions sont détectées. La coopération entre les agents artificiels et l'agent humain prend en compte les éléments subjectifs qui interviennent dans les problèmes complexes de prise de décision.

8.3 COMMUNICATION ET ARCHITECTURES POSSIBLES

8.3.1 CONTRÔLE CENTRALISÉ OU DÉCENTRALISÉ

Le contrôle est dit *contrôle centralisé* si un agent superviseur coordonne l'ensemble des agents et gère l'affectation des tâches à accomplir.
Le superviseur est un agent spécialiste dont le domaine de compétence est la gestion des autres agents. Sa base de connaissances lui permet d'orienter les buts initiaux vers les agents les plus susceptibles de les résoudre compte tenu du contexte. Il peut servir d'arbitre dans la résolution des conflits inter-agents. Il peut gérer l'exclusion mutuelle. Une fois les tâches attribuées, le superviseur n'intervient que pour la surveillance des agents et les agents sont quasi-autonomes.

Dans le cas d'un *contrôle décentralisé,* les décisions vont être prises localement par chaque agent ; les agents vont traiter entre eux par *négociation*.

8.3.2 ARCHITECTURES POSSIBLES

Les agents peuvent communiquer :

- Par *partage de données*; par l'intermédiaire d'une structure de données partagée comprenant tous les éléments nécessaires à la résolution du problème.
- Directement entre eux, par *envoi de messages*, à l'aide d'un protocole de communication. Dans ce cas, les connaissances et les mécanismes de traitement sont distribués dans les agents.

Trois principaux types d'implémentation sont possibles :

8.3.2.1 Architecture de type "tableau noir"

Le *tableau noir* est une structure de données complexe par laquelle va passer toute la communication entre les agents (figure 8.4).

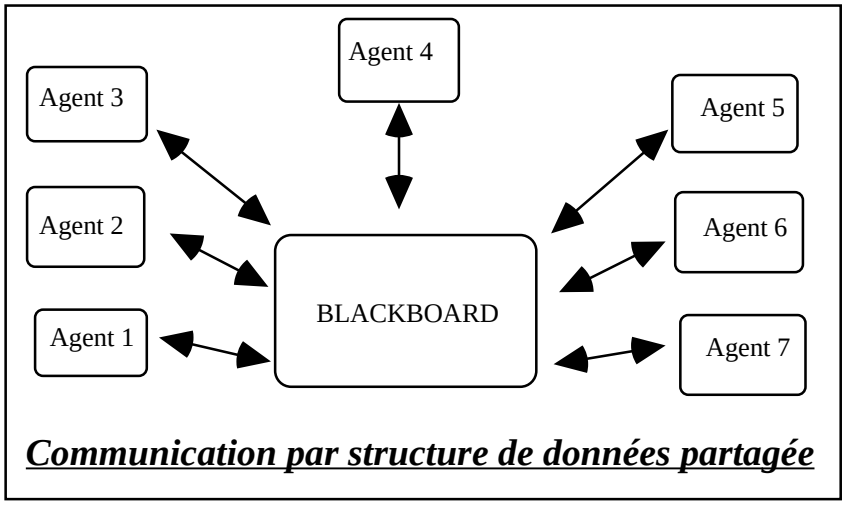


Figure 8.4

Les architectures de type "tableau noir" correspondent en général à des systèmes centralisés. Ces architectures présentent l'avantage de fournir une représentation globale et centralisée de l'état de la résolution et de l'activité du système au prix de l'étranglement des communications via le tableau noir.

Ce mode de communication des agents garantit l'anonymat dans les communications.

8.3.2.2 Architecture avec communication directe entre agents

Les architectures fondées sur *la communication directe entre agents* correspondent en général à des systèmes décentralisés (figure 8.5).
Ce mode de communication permet d'éviter le goulet d'étranglement du tableau noir. En contre-partie, dans un système complètement distribué, la coordination entre agents devient complexe.

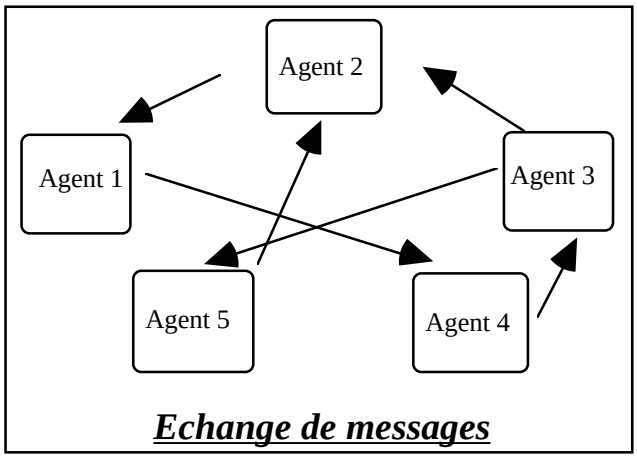


Figure 8.5

Structure élémentaire d'une communication (Occello 93) :

- *L'émetteur* (qui parle ?)
- *Le destinataire* (à qui ?)
- *Le sens sémantique* de la communication (que dit-il)
- *La date* (quand)
- *La durée* et la *fréquence* (Combien de temps et à quelle fréquence)
- *Le type de communication*
- *Le but* (domaine ou comportement)

Quelques implantations de systèmes avec communication inter-agents :

- *Les communautés d'acteurs*; (Hewitt & Liebermann, 84) (Kornfeld & Hewitt, 91) utilisent des messages entre agents dont le comportement est lié aux actions exprimées dans les messages.
- *Les sociétés de spécialistes*, (Gong & Haton, 86) sont caractérisées par une communication sélective entre agents spécialistes.

8.3.2.3 Architecture hybride

Les *architectures hybrides* utilisent à la fois certaines caractéristiques des tableaux noirs et des communications par message.

Par exemple Tigli (Tigli 94), propose comme approche de distinguer clairement les communications nécessaires à la prise de décision du système (basée sur la situation) et les communications nécessaires à l'évolution du système (basée sur les transformations, conséquences des actions des agents). Une partie des communications se fera par partage d'informations, une autre par envois de messages.
Deux types d'agents sont utilisés à cet effet :

- Des *agents d'évaluation* : ils sont chargés de contribuer à l'évaluation de la situation à partir d'une partie des données disponibles au niveau transformationnel.
- Des *agents de transformation* : ils possèdent plusieurs activités possibles selon la transformation choisie par l'agent pour générer ses sorties à partir de ses entrées.

8.4 COMMUNICATION PAR TABLEAU NOIR

8.4.1 GÉNÉRALITÉS

L'*architecture initiale d'un système tableau noir* comprend :

- *un tableau noir*,
- des agents appelés parfois *bases de connaissances*, qui représentent les experts ou spécialistes du domaine,
- *un mécanisme de contrôle* chargé de diriger l'activité des agents en appliquant une stratégie de résolution.

Le tableau noir est dit *hiérarchique* s'il est divisé en niveaux (figure 8.6), chacun correspondant à un type de représentation différent de la connaissance.

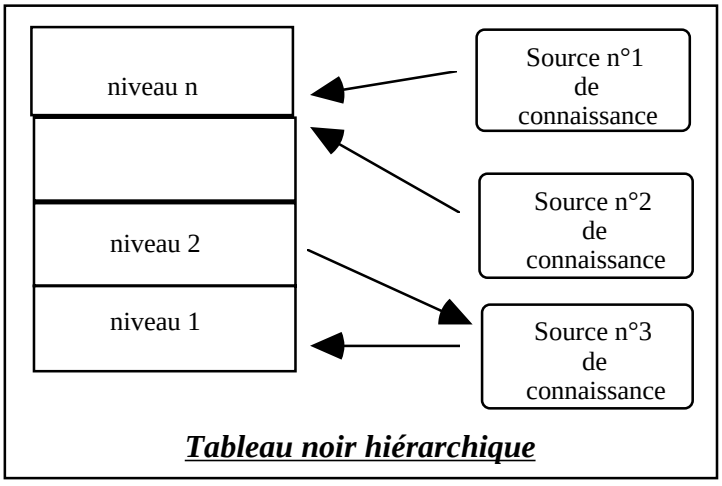


Figure 8.6

Éléments contenus dans un tableau noir
Le tableau noir contient :

- des *éléments de l'espace des solutions*,
- des *données d'entrée*,
- des *solutions partielles*,
- des *alternatives*,
- des *données de contrôle*.

Autre nomenclature :
Les éléments contenus dans le tableau vont aussi conserver les noms provenant de la modélisation des données. Par exemple :

- *hypothèses* si le tableau noir est modélisé en *base de faits* ou de *frame*,
- *objets* si le tableau noir est modélisé en *classes d'objets*,
- *Nœuds* si le tableau noir est modélisé en *structures de données classiques* ou *bases de données relationnelles.*

Caractéristiques principales du modèle :

- Structure homogène des sources de connaissances.
- Les sources de connaissances peuvent lire ou écrire dans le tableau noir.
- Communication indirecte et structurée.
- Les sources de connaissances sont indépendantes les unes des autres. Chaque source ignore la présence des autres.
- La construction de la solution s'établit selon un *processus incrémental.* L'évolution de la solution est perceptible.

8.4.2 LE CONTRÔLE DANS LES SYSTÈMES À TABLEAUX NOIRS

Il existe quatre types classiques de contrôle. Après quelques généralités, nous survolerons successivement les contrôles procédural, hiérarchique, opportuniste et hybride.

8.4.2.1 Cycle élémentaire de base :

- *La détection*

  Elle est fondée sur *les événements* tels que

  modification
  création,
  destruction

  d'objets effectués dans le tableau noir lors de l'exécution d'une base de connaissances.

  Lorsqu'un objet est reconnu, les sources de connaissances créent des *sources de connaissances instanciées SCI.* Les SCI sont placées dans *un agenda* qui tient à jour les possibilités du système.

- *La sélection*

  Elle détermine la prochaine action à exécuter pour faire progresser la résolution du problème. Le contrôle doit faire éventuellement un choix entre des SCI concurrentes.

- *L'exécution*

  Elle détermine le déclenchement de l'instanciation choisie. Les diverses actions possibles correspondent à une mise à jour du tableau noir :

  création de nouveaux objets,
  suppression d'objets anciens,
  modification d'objets anciens.

8.4.2.2 Contrôle procédural

Un *moniteur* régit les accès au tableau noir. Il inscrit dans un agenda les agents dont la condition porte sur un des champs modifiés.

Un *ordonnanceur* détient les heuristiques de contrôle. A partir de l'agenda des agents candidats et de l'état du tableau noir, il lance l'exécution de l'agent le plus prioritaire.
Les connaissances de contrôle sont inscrites dans la structure de contrôle. Un ensemble prédéfini d'heuristiques permet de sélectionner les SCI.

8.4.2.3 Contrôle hiérarchique

Les connaissances de contrôle sont organisées en sources de connaissances hiérarchiques. La communication entre SC est directe dans le sens descendant de la hiérarchie de contrôle et indirect dans le sens ascendant. Les décisions de contrôle ne sont pas explicites.

Les agents sont divisés en trois catégories :

- un *agent "stratégie"*;
- des *agents du domaine*;
- des *agents de contrôle*;

Fonctionnement :
La source stratégie détermine, en fonction des mises à jour sur le tableau noir, les sources de contrôle à activer. Chaque source de contrôle dirige l'action d'un groupe d'agents en fonction des paramètres passés par la source stratégie.

8.4.2.4 Contrôle Opportuniste

Les SC du contrôle et les SC du domaine sont séparées via deux tableaux noirs :

- un *tableau noir du domaine* qui contient les données orientées solution du problème,
- un *tableau noir de contrôle* qui comprend différents niveaux, les niveaux les plus bas représentant les critères de décision utilisés pour le choix.

Un *ordonnanceur* unique décide des activations à mener en considérant les modifications des deux tableaux noirs. Le choix pris par l'ordonnanceur repose sur les données du tableau noir de contrôle.

8.4.2.5 Contrôle Hybride

Le contrôle hybride est un compromis des autres méthodes qui, en général, favorise telle ou telle méthode en vue d'objectifs sur le plan de l'efficacité.

Le système ATOME (Another TOol for developing Multi-Expert systems) représente un contrôle hybride, à la fois global et local, variable selon les phases de résolution d'un problème (Haton 89).

8.4.3 TABLEAUX NOIRS PARTICULIERS

Tableau noir "essentiel" (Tigli 94)

Un *tableau noir essentiel* comprend :

- Une structure de données du domaine simplifiée. L'objectif est de limiter le nombre d'états possibles du système pour permettre une approche plus formelle de la prise de

décision : on ne prend pas en compte le grand nombre des valeurs possibles des données mais seulement un nombre réduit d'états correspondants par exemple à des intervalles de ces valeurs.

- Une structure des données de contrôle. Le problème de la réduction du nombre d'états ne se pose pas, car le nombre des activités possibles d'un agent est en général assez réduit.

Tableau noir "problème"

Il permet la communication et la distribution du contrôle et de la prise de décision parmi les différents types d'agents.

Tableau noir "domaine"

Il contient les actions élémentaires qui résolvent un sous-problème spécifique.

Tableau noir "compatibilité"

Il contient l'ensemble des actions qui satisfont le concept de faisabilité (c'est-à-dire satisfaction des contraintes du domaine) pour l'ensemble des sous-problèmes.

Tableau noir "stratégie"

Il contient, soit une solution globale faisable et cohérente, soit les objectifs contradictoires d'un scénario.

Tableau noir parallèle

Nous allons détailler ce type de tableau noir au paragraphe 8.7.2.

8.5 COMMUNICATION DIRECTE ENTRE AGENTS

8.5.1 LE MODÈLE ACTEUR

Le modèle acteur (Hewitt 73), se fonde sur une communication par envoi de message.

Un acteur

- est une entité informatique non décomposable,
- communique avec d'autres acteurs par envoi de message,
- possède :
    - un environnement,
    - un ensemble d'accointances (agents pouvant l'aider à résoudre une tâche donnée),
    - un ensemble de données et d'actions (script) qui définissent son comportement en fonction du message reçu.

Accointances

Les accointances entre agents décrivent l'organisation statique de la société.

Exemple de règle d'accointances :

Si <condition_de_communication 1> de l'agent X
et <condition_de_communication 2> de l'agent X
et ...
alors <tâche> de l'agent X

Caractéristiques d'un système multi-agents basé sur le modèle acteur

- communication directe entre les agents à l'aide d'un langage commun avec utilisation de protocoles d'interaction,
- répartition des connaissances dans chaque agent leur permettant d'accomplir un traitement local,
- connaissances de ses accointances,
- contrôle généralement distribué.

8.5.2 LE RÉSEAU CONTRACTUEL DE SMITH & DAVIS

Protocole de communication :	• Décomposition de tâches en sous-tâches par un agent particulier appelé *manager* et lancement d'un appel d'offre par celui-ci. • Envoi d'une offre par les agents appelés *contractants* intéressés par le traitement de ces tâches. • Évaluation locale de ces offres par le manager et attribution de la tâche à un agent contractant. • Établissement d'un contrat
Langage de négociation	• Commun à l'ensemble des agents.
Avantages	• Distribution des décisions : contrôle décentralisé dans les agents. • Les agents managers et contractants n'ont pas de rapport hiérarchique entre eux.
Inconvénients	• Structure hiérarchique : le réseau de contrat fixe la décomposition des tâches. • Coût des envois de messages élevé. • Protocole rigide qui ne permet pas de - *procédure d'apprentissage* pour des agents ayant déjà travaillé ensemble, - *procédure de résiliation de contrat*

8.5.3 LE PROTOCOLE D'APPRENTISSAGE DE S. SIAN

Le langage de communication de S.S. Siam (Siam 90) est un langage évolué permettant l'apprentissage entre agents (figure 8.7).

Chaque agent utilise ses informations locales pour en déduire des hypothèses. La coopération avec les autres agents lui permet de modifier le degré de confiance de ses hypothèses. Le mode d'interaction repose sur un tableau noir hiérarchisé : les agents sont connectés à un niveau et les managers aux niveaux supérieurs. Les managers peuvent modifier les informations provenant des niveaux inférieurs.

Le langage de communication comprend 9 primitives de bases :

ASSERT	représente une assertion non modifiable.

PROPOSE	permet de proposer une nouvelle hypothèse.
MODIFY	donne une version modifiée de l'hypothèse précédemment proposée.
AGREED	exprime un accord sur une proposition d'hypothèse.
NOOPINION	exprime une absence d'opinion sur une hypothèse.
DISAGREE	Exprime un désaccord sur une proposition d'hypothèse.
CONFIRM	permet de confirmer l'hypothèse.
ACCEPT	provoque l'acceptation de l'hypothèse.
WITHDRAW	rejette l'hypothèse proposée.

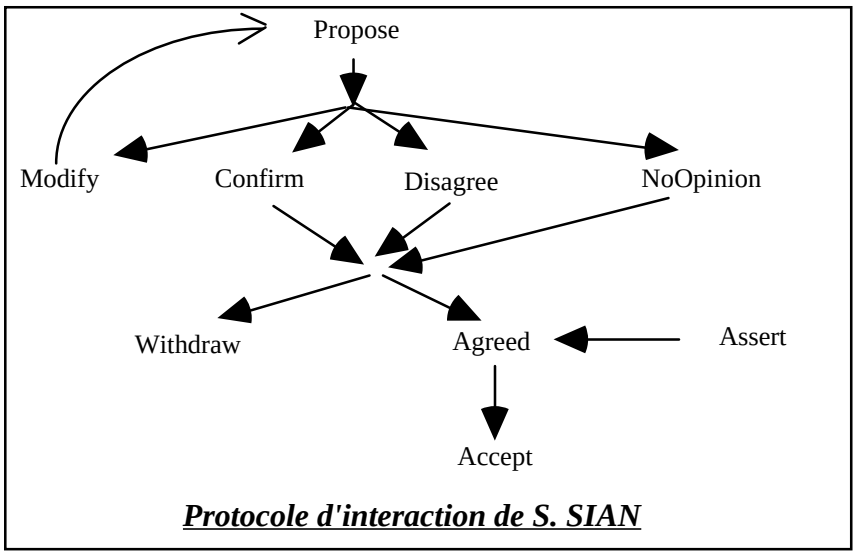


Figure 8.7

Ce protocole a été à l'inspiration de l'architecture de TALISMAN (Stéfanini 93) sur lequel nous reviendrons dans le paragraphe dédié aux systèmes multi-agents en TLN.

8.6 EXEMPLES DE RÉALISATIONS MULTI-AGENTS

8.6.1 EXEMPLE INTRODUCTIF : HEARSAY II

Le premier système faisant intervenir le concept de tableau noir est le système de reconnaissance de la parole HEARSAY-II de Carnegie Mellon University (Erman 1980). Le Tableau Noir contient l'ensemble évolutif de la construction de la solution : les paramètres mesurés, les segments provenant du découpage du signal électrique émis par le microphone, les syllabes, les mots, les tronçons de phrase, les phrases complètes.

Les agents appelés *sources de connaissances* sont des spécialistes :

- identificateurs de signaux électriques (traduction des signaux sonores fournis par l'appareillage d'enregistrement),
- identificateur de phonèmes,
- analyseur lexical,
- analyseur syntaxique.

Architecture générale :

Le tableau noir est un tableau noir hiérarchique (définition au § 8.4.1), l'organisation générale des niveaux étant basée sur la relation "*est composé de*" (ex : une syllabe est composée de phonèmes, un mot est composé de syllabes, une phrase est composée de mots).

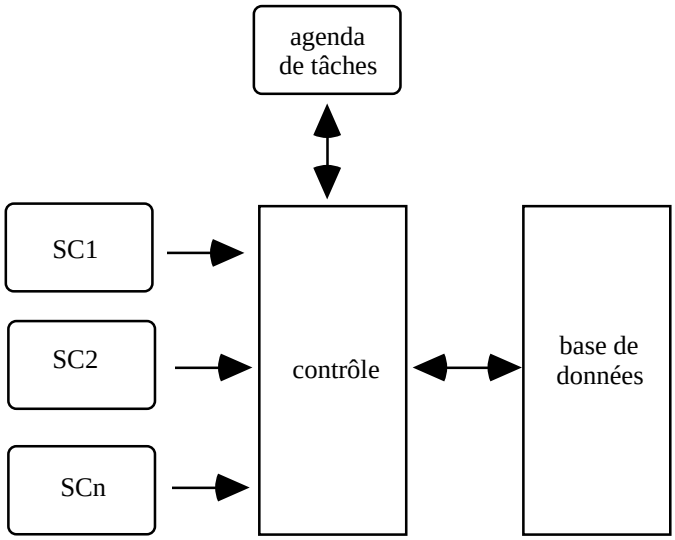


Figure 8.8

Sources de Connaissances :
Hearsay-II est composé de 12 sources de connaissances :

1	SEMANT	génère une requête dans le langage d'accès à la base de données.
2	SEG	numérise le signal, mesure des paramètres, produit une segmentation étiquetée
3	POM	crée des hypothèses de classes de syllabes à partir des segments.
4	MOW	crée des hypothèses de mots à partir des classes de syllabes.
5	WORD-CTL	contrôle le nombre d'hypothèses émises par MOW.
6	WORD-SEQ	crée des hypothèses de séquence de mots.
7	WORD-SEQ-CTL	contrôle le nombre d'hypothèses émises par WORD-SEQ
8	PREDICT	prédit les mots qui suivent ou précèdent une hypothèse de phrase.
9	VERIFY	mesure la cohérence entre les hypothèses de segments et paires de mots consécutifs
10	CONCAT	crée une hypothèse de phrase à partir d'une paire de mots consécutifs vérifiée
11	RPOL	mesure la plausibilité d'une hypothèse
12	STOP	arrête l'exécution lorsque la meilleure interprétation a été trouvée ou quand les ressources sont épuisées.

Mécanisme de contrôle :

Un mécanisme de *contrôle procédural* est chargé de la coopération des agents.

Une dimension rend compte des dépendances entre hypothèses. Des liens, créés lors de l'activation d'une SC, marquent la relation d'hypothèses supports à hypothèse supportée et permettent une gestion cohérente des données du tableau noir.

Chaque hypothèse est pondérée par un *facteur de plausibilité* qui mesure sa cohérence avec, d'une part les données et, d'autre part les sources de connaissances.

L'activation d'une source de connaissances crée, modifie ou supprime des hypothèses dans le tableau noir, crée des liens de dépendance, assigne des coefficients de plausibilité.

A un moment donné du traitement, plusieurs SC sont généralement en compétition. Une seule SC est activée, dont le choix est fonction de la plausibilité du stimulus, du coût d'application de la SC, mais aussi de la réponse probable.

L'algorithme de choix est du type "meilleur d'abord" et utilise les heuristiques suivantes :

1)• *Principe de compétition :*
La meilleure des alternatives en présence est choisie en premier. Lorsque deux hypothèses en concurrence interviennent dans deux stimuli différents, celle de plus grande plausibilité l'emporte.

2)• *Principe de validité:*
Les SC opérant sur les hypothèses les plus plausibles sont choisies en premier. Lorsque deux hypothèses non contradictoires interviennent dans deux stimuli différents, c'est celle de plus forte plausibilité qui l'emporte.

3)• *Principe de signification:*
La SC dont la réponse probable est la plus importante est choisie en premier. L'*importance d'une hypothèse* est une fonction croissante de son niveau d'abstraction.

4)• *Principe de satisfaction du but:*
La SC dont la réponse probable satisfait le but est choisie en premier. La satisfaction de but est une fonction croissante de la *priorité des tâches* répertoriées dans un agenda.

5)• *Principe d'efficacité :*
La SC la plus sûre et dont l'application est la moins coûteuse est choisie en premier.

Pratiquement ces différentes heuristiques sont combinées de manière empirique en une fonction d'évaluation numérique.

L'organisation en SC permet l'intégration de connaissances hétérogènes. L'activation des SC est faite par les données et non par invocation directe. Une SC peut donc ignorer les autres SC, ce qui autorise la possibilité d'un certain parallélisme.

8.6.2 LE SYSTÈME CARAMEL

Ce système (Compréhension Automatique de Récits, Apprentissage et Modélisation des Echanges Langagiers) est destiné à être intégré dans des interfaces vocales Homme-Machine intervenant dans des systèmes tels que les standards téléphoniques,

l'enseignement assisté par ordinateur ou bien la compréhension (indexation automatique, histoires).

CARAMEL est un système multi-agents fondé sur le principe du tableau noir et de gestion de tâches par agenda. (Sabah 90).

Le tableau noir

Le tableau est divisé en trois types de mémoire :

- La *mémoire à court terme* qui contient les phrases formulées par l'utilisateur. Cette mémoire peut être lue par les agents mais ils ne peuvent pas y écrire.
- La *mémoire de travail* qui contient les représentations des différents niveaux (syntaxique, logique, sémantique, ...) et des représentations globales (contexte, ...). Les agents peuvent lire et écrire dans cette mémoire de travail.
- La *mémoire à long terme* contient l'ensemble des connaissances générales stables du système :
  - lexiques,
  - connaissances sémantiques (réseau sous forme de graphes conceptuels),
  - grammaire (environ 450 formes de phrases autorisées),
  - connaissances pragmatiques sous forme de schémas constituant le monde de référence,
  - connaissances spécifiques de chaque processus.

Les différentes sources de connaissances

L'analyse de phrase :

- L'analyseur ANDI (analyseur déterministe, fondée sur une grammaire de cas et une grammaire systémique). L'analyseur construit une représentation syntaxique et sémantique de la phrase en utilisant un regard en avant. La stratégie utilisée est mixte : ascendante et descendante.
- Un système de résolution d'ellipses et d'anaphores,
- Un système de gestion des erreurs lexicales.

La compréhension de texte :

- un système de gestion des personnages qui construit l'image des protagonistes du texte ;
- des systèmes d'interprétations de récits pour obtenir différents points de vue du texte ;
- un système de gestion de plans qui interfère les intentions des personnes en fonction de leurs actions et des connaissances générales sur le monde.

La génération de texte :

- Elle opère sur les représentations obtenues par les modules précédents.

Le contrôle

Le contrôle est centralisé :

- Un superviseur (SIROP) gère la mémoire de travail : il rassemble les résultats fournis, les modifie éventuellement et les insère dans la mémoire de

travail. Le superviseur ordonne les tâches des processus (utilisation de méta-connaissances) et dirige des sous-superviseurs ;
- Les sous-superviseurs déclenchent les processus dans le but de construire la représentation finale demandée. Ils travaillent sur une mémoire de travail personnelle.

La gestion globale des processus s'effectue selon une planification en deux étapes :
- Une planification statique et récursive qui correspond à une décomposition a priori des problèmes en sous-problèmes (raisonnement descendant) ;
- Une planification dynamique qui adapte les résultats de la planification statique en fonction des données à traiter et qui correspond à une coopération par partage de résultats (raisonnement ascendant).

Processus élémentaires et processus complexes

Les *processus élémentaires* ne déclenchent pas directement d'autres processus. Ils travaillent sur une représentation qui leur est donnée et peuvent envoyer un message au superviseur en cas de problème (résolution d'ellipses ou d'anaphores). Ils construisent de nouvelles représentations et sont gérés par des sous-superviseurs.

Les *processus complexes* correspondent à des enchaînement de processus élémentaires.

Principales limites du système
- En cas d'échec d'une tâche, le superviseur ne peut revenir en arrière pour remettre en cause ses choix précédents et recalculer de nouveaux plans.
- Pas de gestion de conflits.

8.6.3 LE SYSTÈME HÉLÈNE

Le système HELENE (Zweigenbaum 89) a pour objectif la compréhension de textes dans un domaine très réduit : celui de compte rendus d'hospitalisation de cancéreux de la thyroïde. La finalité du projet est de constituer un système interrogeable.

Les sources de connaissances d'HELENE se répartissent en :
- des connaissances lexicales et morphologiques,
- des connaissances syntaxiques (utilisation d'une grammaire lexicale fonctionnelle LFG décomposée en deux modules (construction d'une C-structure de constituants et d'une F-structure fonctionelle).
- des connaissances sémantiques,
- des connaissances sur la gestion des événements.

Un superviseur gère les solutions concurrences vues comme des *hypothèses*.

Le contrôle est assuré par un contrôleur à travers un tableau noir ABACAB avec le langage K. Le contrôle est dirigé à la fois par les données et par les buts : les connaissances du contrôle prévoient la coopération des agents. Le contrôle du tableau noir manipule les propriétés des objets du tableau.

8.6.4 LE SYSTÈME TALISMAN

8.6.4.1 Généralités

L'architecture de Talisman (Stefanini 93) est celle d'un système multi-agents gouverné par des lois (SMAGL) destiné à l'étude de la langue écrite (supposée sans erreur).

1) Architecture

Cette architecture repose sur le modèle acteur avec :

- des agents cognitifs qui traitent les problèmes linguistiques éventuellement complexes,
- une communication directe entre agents par envois de message,
- un contrôle par des lois divisées en :
  - lois globales pour la gestion du système,
  - lois locales (pour le contrôle réparti dans chaque agent).

2) Modèle d'agents

Le modèle d'agent de TALISMAN respecte le modèle de l'agent générique défini par Demazeau (Demazeau, 90).

La structure interne dans TALISMAN comporte deux parties :

- Une *partie statique* qui représente les connaissances de l'agent divisée en :
  Mémoire à court terme (point de vue sur la phrase)
  Mémoire à long terme (accointance, structure de données, dictionnaires, grammaires)
  Des buts
  Des compétences
  Des choix
  Des actions
- Une *partie dynamique* qui représente le traitement des connaissances divisée en :
  Des stratégies
  Des mécanismes de raisonnement et de décision.

La structure externe d'un agent représente son comportement (flexible, adaptatif et autonome dans ses actions et ses prises de décision).

3) Découpage des agents

Les agents sont regroupés suivant le niveau (agents *classiques* : lexical ou syntaxique) et suivant les phénomènes complexes à traiter (agents *transversaux* : coordination, négation, …).

Agent	But	Connaissances	Compétences	Stratégies
Pret	Transforme un fichier ou une phrase en une suite de formes pouvant être analysées	Ensemble de règles en vue des traitements morpho-graphique et morpho-syntaxique	Prétraitement morpho-graphique et morpho-syntaxique	Repérage du marqueur de fin de phrase puis en séquence : morphologie et syntaxe

Morph	Analyse morphologique	Dictionnaire du français courant (DELAS), dictionnaires de spécialité	Analyseur morphologique flexionnel, Désambiguïsation à l'aide de règles linguistiques Regroupement de morphèmes discontinus	Choix du dictionnaire Analyse morphologique flexionnelle Désambiguïsation à l'aide de règles linguistiques
Segm	Segmente la phrase en propositions	Liste des marqueurs de limite de propositions (introducteurs de subordonnées, conjonctions de coordination, virgules, verbes conjugués)	Compteur de verbes, détection des propositions principales et subordonnées, construction des arbres (phrase en propositions) détection d'incises	Prédiction sur le nombre de proportions, Segmentation et découpage en niveaux. Deux stratégies particulières (coordination et certaines incises)
Synt	Trouve les bonnes analyses syntaxiques d'une proposition	Dictionnaires des indicateurs de structures lexicaux (ISL) et de structures grammaticaux (ISG), Liste de prépositions introduisant des circonstanciels, grammaire de base	Analyse syntaxique en constituants de la proposition, Désambiguïsation par vérification des accords, Détection des ISG pour la reconnaissance des syntagmes et des ISL pour leur rôle fonctionnel. Et la reconnaissance de certains circonstanciels	Prédiction basée sur les ISL et les ISG. Appel à l'analyseur d'Earley. Détermination du caractère des compléments. Détermination de la structure fonctionnelle.
Transf	Détecte et annote les phrases impératives dans le but d'obtenir une forme canonique	Module d'inversion du sujet, liste de formes introduisant une interrogative, une phrase clivée, une impersonnelle	Détection et transformation de certaines questions totales, questions partielles, impératives, phrases clivées et impersonnelles.	Recherche et normalisation de certaines questions totales, questions partielles, impératives, phrases clivées et impersonnelles.
Ellip	En cours de développement (Warren 98)			
Coord	En cours de développement			
Néga	En cours de développement			

Tableau récapitulatif des agents de TALISMAN

4) TALISMAN II 98 / TALISMAN I 93

La version 98 de TALISMAN (Warren 98) présente des modifications importantes pour l'amélioration des performances du système, la finalité pouvant être une meilleure gestion des conflits dans ce type d'architecture multi-agents.

Au niveau des modifications simples et ajouts, citons :

- Amélioration des compétences de désambiguïsation de l'agent MORPH par un développement de règles linguistiques contextuelles de désambiguïsation.
- Transformation de l'agent SYNT par remplacement de l'analyseur d'Earley par un analyseur permettant l'utilisation d'heuristiques

- Augmentation des stratégies disponibles pour l'agent SEGM.
- Etude des interactions de COORD avec les autres agents.
- Amélioration de la modélisation des agents TRANSF, ELLIP, et NEGA.
- Première spécification d'un agent ANAPH expert en résolution d'anaphores.

Par ailleurs, partant de l'idée que toute forme de contrôle centralisé est impartiale, le concept de la gestion par des lois est abandonné.

8.6.4.2 Le module Synt et la vérification des accords

Le module SYNT, faisant de la vérification des accords, méritait bien un petit examen !

La vérification des accords est faite afin d'éliminer plus rapidement les solutions parasites.

Le texte traité est supposé correct. De ce fait, sans état d'âme, on peut dans certaines circonstances rejeter des solutions.

Les vérifications mettant en jeux des nominaux et/ou des déterminants s'effectuent sur le genre et le nombre.

Les vérifications des accords sur des préverbaux et/ou des verbes sont réalisées en fonction de la personne.

En cas d'échec, on rejette la solution.

> Exemple :
> Dans *ce poste*
> *ce* est masculin,
> *poste* peut être féminin (si *poste* = le service des postes)
> ou masculin (si *poste* = l'endroit).
> Le poste féminin va être rejeté par vérification des accords (groupement D+F).

Comme nous l'avons vu, la mentalité est différente de celle du domaine de la correction d'erreurs qui va mettre éventuellement en doute aussi le *ce*. Pour un rédacteur faisant beaucoup de fautes de suppression, on ne pourra pas exclure que *ce* est en réalité *cette* et que donc *poste* est bien celle au féminin ! En réalité, il faudra chercher plus loin dans la phrase la désambiguïsation et on retombe sur la méthode des structures du chapitre VII.

8.6.4.3 Conclusion

Les travaux de Marie-Hélène Stéfanini m'ont été précieux en début de thèse sur plusieurs plans :

- celui de la problématique de l'analyse d'une langue naturelle,
- celui de la compréhension de la notion de système multi-agents et des spécificités d'un système multi-agents appliqué au TALN,
- celui de la communication par message et du type de coopération qu'il entraîne.

La problématique de la correction d'erreurs incluant celle de l'analyse des langues naturelles est donc plus large. En conséquence, l'ajout de nouvelles exigences ont entraîné pour CELINE des choix fondamentalement différents de ceux de TALISMAN.

8.7 TABLEAU NOIR ET CELINE

8.7.1 SYSTÈME MULTI-AGENTS ET CELINE

8.7.1.1 Introduction

CELINE utilise des modules pré-existants et des agents développés spécialement. Les modules préexistant ne sont pas du tout développés en vue d'un système multi-agents. Il faut donc que ces modules soient interfacés, l'interface devant assurer au minimum deux fonctions (figure 8.9) :

- la mise en correspondance des traits manipulés dans CELINE et ceux manipulés dans le module,
- la gestion de la communication extérieure et intérieure.

Pour la suite, lorsque nous parlerons d'un *module X* en le désignant par le label *agent X*, nous désignerons l'ensemble module plus interface.

Dans le cas d'une communication par message et d'un contrôle réparti, il faut de plus que l'interface assure les fonctions correspondantes.

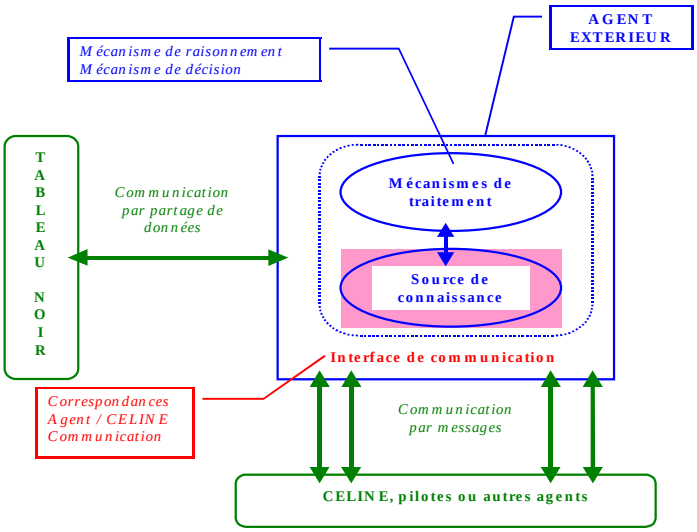


Figure 8.9

Trois questions se posent :

- Les modules ainsi interfacés méritent-ils le qualificatif d'agents ?
- Le tableau structure de données et le mode de fonctionnement proposé répondent-ils aux critères des tableaux noirs ?
- Parmi les variantes de contrôle quel est celui utilisé ?

Examinons successivement chacun de ses points.

8.7.1.2 Les agents

- Au chapitre suivant sur CELINE nous donnerons pour quelques agents une réponse positionnant l'agent par rapport à la définition d'agents génériques de Damereau :

Agent	But	Connaissances	Compétences	Stratégies

- Provisoirement, posons nous la question vis-à-vis des modules traditionnels du type générateur morphologique, analyseur syntaxique, modèles de Markov. Ils passent les critères de bases en repoussant la prise en compte de l'aspect *agent* au niveau de l'interface assurant les échanges entre agents et le contrôle choisi.

	PILAF (générateur)	PILCHART	MARKOV
Un *agent* est • une *entité autonome*, • qui poursuit un *objectif individuel*, • qui est apte	⇒ Oui ⇒ Génération	⇒ Oui ⇒ Analyse	⇒ Oui ⇒ Calcul matrices et proposition de catégories
• à *agir sur l'environnement du système* et/ou • à *interagir avec les autres agents*	⇒ Fournit des formes textuelles prise en compte par les autres agents. ⇒ Peut travailler directement avec le vérificateur des accords	⇒ Fournit les arêtes correspondantes à l'analyse partielle ou totale) lorsqu'une solution est trouvée pour la phrase.	⇒ Fournit des catégories morphologiques
• qui ne dispose que d'une *représentation évolutive* de cet environnement, • qui peut *percevoir les autres agents*	⇒ Accès au tableau *structure de données* ⇒ Demande et réponse directe (par exemple avec l'agent vérificateur des accords)	⇒ Accès au tableau *structure de données*	⇒ Accès au tableau *structure de données* ⇒ Demande et réponse directe (par exemple avec un analyseur syntaxique)
La *structure d'un agent* comprend : • des *connaissances* (connues initialement ou acquises par l'agent)	⇒ Racines, désinences, règles	⇒ Règles grammaticales	⇒ Matrices de transitions
• des *buts*	⇒ Générer la forme correcte	⇒ Dresser la LAT et fournir les structures	⇒ Proposer des catégories ⇒ Calculer des probabilités de parcours
• des *actions* (sur l'environnement ou sur lui même).	⇒ Intervient sur le tableau de données	⇒ Intervient sur le tableau de données	⇒ Intervient sur le tableau de données et modifie ses matrices
Les *mécanismes de traitement d'un agent* se divisent en : • *mécanismes de raisonnement d'un agent* qui *construisent* les différentes actions possibles en fonction des buts et des connaissances de l'agent.	Agent du type *réactif* ou démon actif scrutant le tableau noir	Agent du type *réactif* ou démon actif scrutant le tableau noir	Agent du type *réactif* Ou démon actif scrutant le tableau noir
• *mécanismes de décision d'un agent* qui en fonction des buts et	⇒ Interprétation des marques de validité du tableau *structure*	⇒ Interprétation des marques de validité du tableau *structure de*	⇒ Interprétation des marques de validité du tableau *structure de*

des hypothèses *choisissent* la ou les actions les plus appropriées.	*de données.*	*données.*	*données.*

8.7.1.3 La structure de données et le tableau noir

La structure de données et le mode de fonctionnement proposés pour CELINE correspondent-ils aux critères de définition d'un tableau noir ? :

Premier élément de réponse :
Entre les chapitre 2 et 4, nous avons défini la structure de données ainsi que les interventions (« humaines ») des agents travaillant sur cette structure de données.

- Exemple du puzzle

  > Exemple de Nii (Nii 86) sur la construction collective d'un puzzle servant d'introduction justificative au concept de tableau noir.
  > Un groupe de personnes doit construire un puzzle. Chacune possède un jeu de pièces. A chaque nouvelle pièce ajustée, chacun peut analyser la situation et décider s'il peut ou non poser une pièce. Chaque personne réagit à une modification du puzzle et travaille sans échange direct avec les autres personnes.

- Nous retrouvons exactement l'équivalent de l'exemple de la classe faisant un exercice collectif de correction au tableau noir

  > Un groupe d'élèves doit corriger une phrase modélisée au tableau sous la forme de la structure proposée.. Chacun possède des connaissances. A chaque ajout d'un des élèves, chacun peut analyser la situation et décider s'il peut en déduire un nouvel élément acceptation, refus ou proposition. Les élèves ne parlent pas entre eux (mais si ! c'est possible !) et travaillent uniquement en regardant le tableau. Le maître d'école inscrit lui-même les réponses proposées.

Deuxième réponse plus technique :
Le tableau *structure de données* sera appelé *TSD* dans le tableau ci-dessous.

Comparaison de la définition d'un blackboard et du tableau *structure de données* de CELINE	
Tableau noir hiérarchique	Le TSD est divisé en niveaux (forme textuelle, catégorie morphologique, variables morphologiques, structures etc.).
Le tableau noir contient :	
• des *éléments de l'espace des solutions*,	⇒ Les formes textuelles finales de la phrase corrigée sont dans le TSD.
• des *données d'entrée*,	⇒ Les formes textuelles de la phrase à traiter sont dans le TSD
• des *solutions partielles*,	⇒ Les lignes du TSD sont complétées petit à petit. Les structures grammaticales sont construites aussi au fur et à mesure. Etc.
• des *alternatives*,	⇒ Le TSD renferme toutes les solutions concurrentes sur les divers niveaux : catégories multiples d'une même forme textuelle, formes textuelles solutions concurrentes d'une forme inconnue, Charts multiples etc.
• des données de contrôle	⇒ Les marques de validité, les poids des structures etc.

• La construction de la solution s'établit selon un processus incrémental.	⇒ Des lignes sont complétées, d'autres lignes sont rajoutées ; petit à petit certaines solutions sont éliminées, etc.
• L'évolution de la solution est perceptible.	⇒ Les marques de validité, les structures non terminales etc.

8.7.1.4 Le contrôle

Les choix sur le contrôle ont été guidés par trois idées fortes :

1. La première idée est d'avoir le moins de contrôle possible.
   - En effet, qui dit contrôle dit stratégie, et qui dit stratégie a du mal a se débarrasser du déterminisme. Les problèmes linguistiques à résoudre pouvant varier à l'infini, une stratégie figée (par exemple planification …) a peu de chance d'être satisfaisante.
   - Le corollaire est que le contrôle va finalement être effectué par les données. Bien entendu, il faut quand même qu'un (ou des) agents interprète(nt) ces données.

2. La deuxième idée est qu'une communication directe entre agents pose des problèmes lors d'ajout d'agents. En effet, à la limite, il faut répercuter son existence dans les accointances de tous les autres agents, ce qui demande de les modifier (trop coûteux pour un grand nombre d'agents). Le système du tableau noir supprime cet inconvénient puisque chaque agent intervient d'une façon indépendante.

3. Caromel (Caromel 91) défini un système parallèle comme un système où plusieurs activités se produisent simultanément. C'est exactement ce que nous voulons que nos agents fassent et nous avons déjà dit qu'ils doivent pouvoir intervenir en parallèle sur la structure de données. Dans le cas d'un grand nombre d'agents nous obtenons alors une situation très difficile à gérer.

L'ensemble de ces éléments nous a conduit à faire un choix parmi les deux modes d'introduction du parallélisme dans les architectures tableau noir (Corkill 89) :

- Les *tableaux noirs distribués*. Dans cette architecture , chaque tableau noir est situé sur un processeur. Ces tableaux noirs du type *séquentiels autonomes* échangent des données. Les agents sont exécutés en séquentiel ou en multi-programmation.
- Les *tableaux noirs parallèles*. Dans cette architecture, les agents interagissent dans un espace partagé. Les agents sont eux distribués sur différentes machines. Nous retrouvons là notre conception du système de correction des erreurs. Nous allons donc retenir cette solution et la développer dans le paragraphe suivant.

8.7.2 TABLEAU NOIR PARALLÈLE

Les composants principaux d'une architecture tableau noir sont les agents, la structure de données et le mécanisme de contrôle. L'introduction du parallélisme va se répercuter sur chaque composant. Dans cette partie nous nous intéressons uniquement au travail sur la structure de données et non à la gestion des agents.

8.7.2.1 Les agents

Le fonctionnement interne des agents peut utiliser un *parallélisme intra-agents*. Cela correspond à un choix de développement et n'est absolument pas une obligation.

Par contre le *parallélisme inter-agents* des tableaux noirs parallèles demande de modifier les mécanismes des systèmes séquentiels au niveau du contrôle et de la gestion de la structure de données.

8.7.2.2 Le contrôle

Toutes les modifications du tableau noir doivent être prises en compte. Deux grands choix sont possibles :

- Activation dynamique des agents,
- Signal d'attention sur une information qui les intéresse.

Par rapport à un système séquentiel, les nouvelles contraintes possibles sont les suivantes :

- Contraintes d'accès (interdiction de l'exécution simultanée d'agents utilisant les mêmes données).
- Contraintes d'ordonnancement (ordre de déclenchement ou interdépendance).
- Contrainte d'exclusion : agents en compétition à élection exclusive. Par exemple on peut décider de travailler avec une grammaire de dépendances ou avec une grammaire de constituants. Des couples analyseurs et modules de vérification des accords seront possibles et d'autres exclus.

Le contrôle peut être :

- centralisé (un seul agent, ce qui pose le problème d'un goulet d'étranglement si la coopération et donc le nombre de communications sont élevés),
- décentralisé (plusieurs agents par duplication),
- délocalisé avec suppression de l'unité de contrôle. Dans ce cas le contrôle est laissé aux données : l'interprétation des données par les différents agents va amener à une évolution de la structure de données vers la (ou les) solution(s) finale(s). Ce type de contrôle est celui que nous avons déjà souhaité au paragraphe 8.7.1.4.

8.7.2.3 La structure de données

Pour permettre une meilleure activité nous supposerons tout d'abord que les contraintes d'accès à la structure de données ne s'applique qu'aux écritures, les lectures restant possibles à tout instant.

Les contraintes d'accès peuvent être réglées par des sémaphores. Par ailleurs, nous avons déjà envisagé d'utiliser une implantation du tableau noir sous la forme de tables d'un SGBD.

Quelle que soit la solution physique envisagée, le problème dit de la *cohérence sémantique* se pose (Corkill 89) : il faut qu'entre le moment où un agent lit des informations dans le tableau noir et le moment ou il y reporte le résultat de ses travaux, les informations prises en compte ne soient pas modifiées.

Quatre méthodes sont utilisées pour la synchronisation (Fennel 77) repris par (Occello 93) :

- Technique du *verrouillage* (*Locking*). Cette technique porte sur le tableau : les objets ou groupe d'objets du tableau noir utilisés par un agent sont verrouillés. L'inconvénient évident est la possibilité de *deadlock* ou bien des temps d'attente trop longs dans le cas d'un grand nombre de données indisponibles. Dans le cas de notre système comportant beaucoup d'agents et notre choix de ne pas avoir un contrôle strict, cette solution est à rejeter devant le risque élevé de voir tous les agents verrouiller une partie du tableau.

- Technique du *marquage* (*Tagging*). Cette technique porte sur les agents. Les données lues (*contexte local*) dans le tableau noir par un agent A1 sont *marquées* par l'agent qui les utilisent. Un autre agent A2 modifiant un ou des champs du contexte local de A1 va être prioritaire et va pouvoir porter ses modifications. Un message est alors envoyé à l'agent A1 qui va pouvoir poursuivre son travail si les anciennes valeurs des champs modifiés n'ont pas été encore prises en compte, abandonner son travail ou reprendre à zéro avec le nouveau contexte. Cette solution est intéressante pour des agents effectuant un travail long. Dans notre cas, nous rejetterons cette solution car le travail d'envoi des messages serait plus coûteux en ressources et en temps que le travail de l'agent lui-même (rappel : le travail en réseau rend très pénalisant les temps de communication).

- Technique de la *supposition* (*Assumption*). Avant de porter une modification dans le tableau noir, l'agent vérifie si le contexte n'a pas été modifié et s'il correspond toujours au contexte local précédemment lu. Dans le cas contraire son travail est abandonné. Cette solution a été partiellement retenue pour CELINE qui va l'appliquer en se basant sur les marques de validité. On peut distinguer trois cas :
    1. Si la ligne du tableau a été rejetée : le travail est abandonné.
    2. Si un autre agent a déjà amené une réponse identique pour les mêmes champs alors on ajoute la marque de validité de l'agent.
    3. Si par contre un autre agent a déjà amené une réponse différente pour les mêmes champs alors on génère une ligne supplémentaire au tableau. Selon les cas, on porte la marque de validité *refus dans l'ancienne ligne*. Le nombre de solutions concurrentes et le degré d'ambiguïté croissent.

- Technique supposant l'inutilité de la garantie de cohérence sémantique (*Fonctionnaly / accurate Cooperation*). Dans ce cas les agents amènent leurs modifications sans se soucier des modifications intermédiaires inscrites dans le tableau noir. Chaque agent est supposé faire de son mieux avec les données dont il dispose. Dans le cas où les modifications amenées par l'agent ne permettent pas d'obtenir la solution finale, les données modifiées vont avoir un statut d'hypothèses pour les autres agents qui vont repartir en mode correction. Cette solution est aussi partiellement retenue pour CELINE qui, dans certaines circonstances, va rajouter des lignes au tableau sans se soucier des résultats intermédiaires obtenus. Ajoutons que

l'antériorité d'une action sur une autre n'est pas prise en compte car cette antériorité peut provenir par exemple d'une communication réseau passagèrement ralentie.

8.7.2.4 Prototype et parallélisme

Nous avons aussi vu apparaître le parallélisme avec ses techniques et ses problèmes particuliers. Dans le cadre de cette thèse (sur la correction des erreurs !), nous avons choisi de ne pas aborder les problèmes du parallélisme liés à l'architecture machine (grain, couplage, système fortement ou faiblement couplé etc.).

De la même façon, le système distribué des agents lexicaux pose le problème des pannes machines ou des pannes sur le réseau.

La création d'un prototype orienté correction d'erreurs n'exige pas une étude complète sur ces points. Par contre tout développement ultérieur en vue d'un système réel demanderait d'étudier ces deux dimensions.

Nous supposerons donc provisoirement que toutes les demandes en processeurs sont satisfaites et nous nous reposerons sur les systèmes d'exploitation :

1. Dans le cas de machines multi-processeurs nous laisserons au système le choix de l'association processus et processeur (*placement des tâches*).
2. Dans le cas d'une machine mono-processeur, nous admettrons que plusieurs processus peuvent cohabiter sur ce même processeur qui les sert alternativement en temps partagé (méthode de la *multi-programmation*). Cette *méthode simule le traitement parallèle* tout en accroissant le temps d'exécution par les changements de contexte qu'elle impose. Il s'agit du mode normal de fonctionnement dans le cas d'UNIX.
3. Pour des sites lointains nous nous ramènerons au cas précédent, le réseau (ETHERNET) assurant les protocoles de communication. Pas de panne : tous nos agents sont disponibles. Nous pouvons remarquer que l'éventuelle présence d'agents concurrents, implantés sur des machines différentes, permettrait au système de faire face à la panne d'une machine.

8.7.3 ARCHITECTURE FINALE DE CELINE

Le chapitre sur CELINE va nous présenter son architecture complète que nous pouvons qualifier d'*hybride* puisque mélangeant un contrôle centralisé (pour la gestion des agents) et un tableau noir parallèle (pour le travail sur le tableau noir) :

- Un *superviseur* CELINE s'appuyant sur une architecture hiérarchique d'agents pilotes de secteurs d'activités (activité lexicale, activité syntaxique, activité de vérification des accords etc.) va pour l'essentiel :
    1. initialiser la structure de données,
    2. activer les pilotes,
    3. détecter la fin du travail (réussite ou échec),
    4. stopper les pilotes encore en activité.
    5. Avant d'effacer le tableau noir, le superviseur va déclencher à partir de son contenu des mises à jour dans la connaissance du rédacteur (par activation de l'agent MAEC) et dans la connaissance du système (par l'activation de l'agent DSR).

- Ces pilotes, agents cognitifs, sont chargés de l'activation des agents de travail de leur domaine (bases de connaissances). Le superviseur et les pilotes vont communiquer directement entre eux par messages.

- Les agents de travail eux vont intervenir directement sur le tableau noir parallèle (§ 8.7.2). Ces agents peuvent être cognitifs ou réactifs. Ils vont fonctionner en démon scrutant le tableau et réagissant lorsque leur analyse leur montre une possibilité d'intervention.

8.8 CONCLUSION

Tout au long des chapitres précédents, nous avons établi un cahier des charges pour CELINE en partant des besoins de la correction des erreurs.

Dans la première partie de ce chapitre nous avons regardé la taxinomie et le mode de fonctionnement des systèmes multi-agents et tout particulièrement ceux à base de tableaux noirs. Parmi les systèmes tableaux noirs, nous avons plus spécialement détaillé les tableaux noirs parallèles.

La comparaison du cahier des charges et des SMA nous a montré une identité de point de vue et il est donc raisonnable d'envisager un système de correction d'erreurs comme un système multi-agents. Le prochain chapitre sur CELINE se placera donc résolument dans cette optique.

9. LE SYSTÈME CÉLINE

Finalité
La finalité de ce chapitre est une présentation globale synthétique de CELINE (**CELINE** = **C**orr**E**cteur **L**ex**I**co-sy**N**taxiqu**E)** renforcée par des compléments mais sans reprendre les points de fonctionnement déjà exposés dans les précédents chapitres. L'objectif prioritaire est de définir une vue d'ensemble cohérente du fonctionnement global visé.

Tout au long des chapitres 1 à 8 nous avons défini le correcteur orthographique que nous souhaitions réaliser par le choix des options fondamentales. Rappelons que dans le chapitre d'introduction, le tableau du paragraphe 1.3.2 présente un résumé du cahier de charge et du spécifique sur CELINE établi chapitre par chapitre.

Dans ce chapitre nous ne détaillerons donc pas les différents agents dont les définitions sont réparties au fil des chapitres.

Résumé
Dans un paragraphe d'introduction nous allons rappeler certains points généraux de la problématique justificatifs de certains compléments.

Dans une deuxième partie, nous présenterons des généralités sur l'architecture *hiérarchique pseudo-pyramidale hybride* du système, les modes de communications et pour terminer nous établirons une classification des agents.

Après des généralités par le contexte de développement de CELINE et son paramétrage linguistique, une partie intermédiaire permettra de comprendre le passage d'un texte formaté à une structure de données représentative de ce texte et dont une partie permet l'initialisation du blackboard.

Nous pourrons alors entreprendre la description du fonctionnement global par l'étude du contrôle au travers du fonctionnement du superviseur et des pilotes. S'intéressant au pilote des activités lexicales, nous découvrirons une méthode de recherche lexicale basée sur un *lexico-thésaurus*, méthode servant aussi bien à optimiser la recherche du bon lexique qu'à définir les domaines d'un utilisateur.

Une dernière partie récapitulera les agents développés ou en cours de développement et présentera une extension de la classification des agents lexicaux.

9.1 INTRODUCTION

Problématique par rapport à la linguistique

La correction nécessite la mise en œuvre d'informations linguistiques d'origines très variées :

- phonétiques pour tenir compte des variations individuelles (accent, type de prononciation...) permettant de comprendre l'origine de certaines fautes et par là même de sélectionner les corrections les plus vraisemblables;
- lexicales pour chaque mot, unité de structuration de la phrase ;
- syntaxiques pour traduire la structure d'une phrase en fonction des règles grammaticales du langage ;
- sémantiques pour obtenir la signification des mots et leurs liens ;

- pragmatiques pour transmettre le contexte de la phrase et l'historique du discours (bien que loin du domaine orthographique, la pragmatique aide la sémantique et peut donc sembler nécessaire).

La codification de ces informations se fait par l'intermédiaire de lexiques, de grammaires, de graphes de transition … mais pour le moment l'état de l'art ne permet pas de disposer de l'ensemble des informations (phonétiques à pragmatiques) qui seraient pourtant indispensables. Par exemple :

1) On ne sait pas déterminer une grammaire suffisamment complète pour une langue naturelle infinie. L'analyse syntaxique touche là une de ses limites.

2) On ne dispose pas de formalismes faisant l'unanimité pour la codification de la sémantique. De plus devant l'énormité de la tâche, les réalisations ne sont que partielles et limitées à des sous-langages finis.

3) La pragmatique, récente, s'applique dans des cas particuliers tel ceux des dialogues homme-machine finalisés ou encore sous la forme d'une pragmatique textuelle dans des domaines précis.

Au chapitre un sur la problématique générale de la correction d'erreurs nous avons rappelé que la correction d'erreurs ne peut pas être efficace si on envisage un ordonnancement séquentiel des outils classiques : d'abord l'analyse morphologique, puis l'analyse syntaxique, puis .... Il faut rendre quasiment simultané le traitement des différents niveaux (lexical, syntaxique, sémantique).

Cette intégration des actions aux différents niveaux peut être obtenue de diverses manières à travers d'une part les bases de connaissances et d'autre part les outils :

- Une des voies possibles est la création de lexiques comportant l'ensemble des traits utiles avec, en parallèle, le développement d'outils d'un type nouveau permettant la manipulation de ces traits complexes. Or la majorité des équipes de recherche travaille dans un domaine en général assez étroit et doit ou bien se limiter à ce domaine ou bien se servir de bases de connaissances développées par d'autres équipes avec souvent des formalismes différents ce qui rend un interfaçage direct peu probable.

- Une solution d'intégration de données peut être par exemple la fusion de lexiques. Ce type d'opération est en soi très coûteux et la mise à disposition opérationnelle est lente. De plus, les outils doivent de toute façon être au minimum retouchés et la plus part du temps complètement réécrits.

- Une autre méthode va consister à conserver des lexiques "simples", indépendants, avec, pour chacun d'entre eux, un nombre de traits réduits. L'intégration des différents niveaux s'obtient alors par une faible granularité au niveau des données traitées et un morcellement de la tâche globale. Martin KAY lors de sa conférence "*Event sementics and chart generation*" au GETA en juin 1994 faisait part d'une "dernière tendance de recherche" aux USA avec un retour vers une codification des connaissances par niveaux séparés (lexique, syntaxe, sémantique), tendance à laquelle il adhérait. On trouve là une justification pratique d'une approche multi-agents : chaque agent garde les bases de connaissances spécifiques de son domaine d'expertise et va travailler en parallèle avec les autres agents sur une petite partie de la phrase ou du texte à traiter.

Dans ces conditions, sur le plan du génie logiciel, on obtient une souplesse de développement convenable et comparable à celle de la classique décomposition modulaire. Chaque équipe peut continuer à faire évoluer ses bases de connaissances et ses agents d'une manière quasi indépendante, la seule contrainte obligatoire étant alors le respect du mode de communication choisi et des protocoles s'y rattachant.

Cette indépendance inter-agents, réelle dans le cas des agents réactifs, trouve pourtant ses limites avec les agents cognitifs dont l'intelligence dépend des connaissances sociales, avec pour les plus évolués, la connaissance du modèle explicite des autres agents et des capacités de raisonnement comprenant la manipulation des plans des autres agents.

Des remarques ci-dessus nous tirerons une exigence au niveau du développement logiciel : les outils de bases faisant appel à des connaissances doivent pouvoir être utilisés isolément et indépendamment du système CELINE.

Nous souhaitons approcher un système robuste permettant une correction automatique au moins dans le cas d'un rédacteur connu. Nous devons donc potentiellement pouvoir couvrir n'importe quel domaine et ne pas adopter de stratégie a priori du type première(s) solution(s) trouvée(s). Seule la connaissance d'un rédacteur particulier et de son MLPS va nous permettre de mettre en place des mécanismes de rejet suffisants des solutions concurrentes.

Problématique par rapport au parallélisme

Le parallélisme permet d'exécuter simultanément différentes tâches, par exemple une analyse lexicale sur un mot de la phrase, une correction sur un autre mot (préalablement) non reconnu et une analyse syntaxique partielle sur un tronçon de phrase.

Le parallélisme trouve ses limites dans le nombre de processeurs de la machine hôte ou dans la gestion et la fiabilité des échanges si on utilise plusieurs machines en réseau. De plus, les machines permettant le parallélisme sont multi-tâches et de ce fait, la charge de travail due aux autres utilisateurs peut ralentir considérablement l'activité apparente ; tout cela étant aggravé par le fait que la priorité allouée à des processus de correction d'erreurs risque bien d'être faible.

Si on utilise plusieurs machines, on bénéficie d'une possibilité de répartition de la masse importante des connaissances. Par contre, les échanges et les transferts de données ralentissent le fonctionnement.

Dans tous les cas on se trouve confronté à de délicats problèmes de synchronisation des différents processus.

Bien que nous ne gérions pas vraiment le parallélisme (remarque du § VIII.7.2.4 sur les performances supposées suffisantes du système) et le nombre de processus possibles, à titre d'expérimentation, nous avons supposé que nous avions droit à N=30 agents actifs en même temps se répartissant en :

- 4 agents toujours actifs ou pouvant l'être : *CELINE, BRUNO, MAEC et DSR*

- 10 agents dans le domaine lexical (pilote *PAL*, agent *TJRP*, agent *CCFL*, agent *PILAF*, l'agent lexical du MLPS et 5 autres agents à gérer).
- 10 agents dans le domaine syntaxique (pilote *PAS*, analyseurs *PILCHART* et *TOMITA*, agent de proposition statistique *PSCM*, agent de correction contextuelle).
- 6 agents dans le domaine de la correction des erreurs : (pilote *PAVA*, agent de détection et correction des fautes d'accord *DCFA*).

L'introduction d'une limitation du nombre d'agents permet de mettre en place la stratégie d'optimisation de choix des agents.

9.2 GÉNÉRALITÉS SUR L'ARCHITECTURE

9.2.1 LE CONTRÔLE

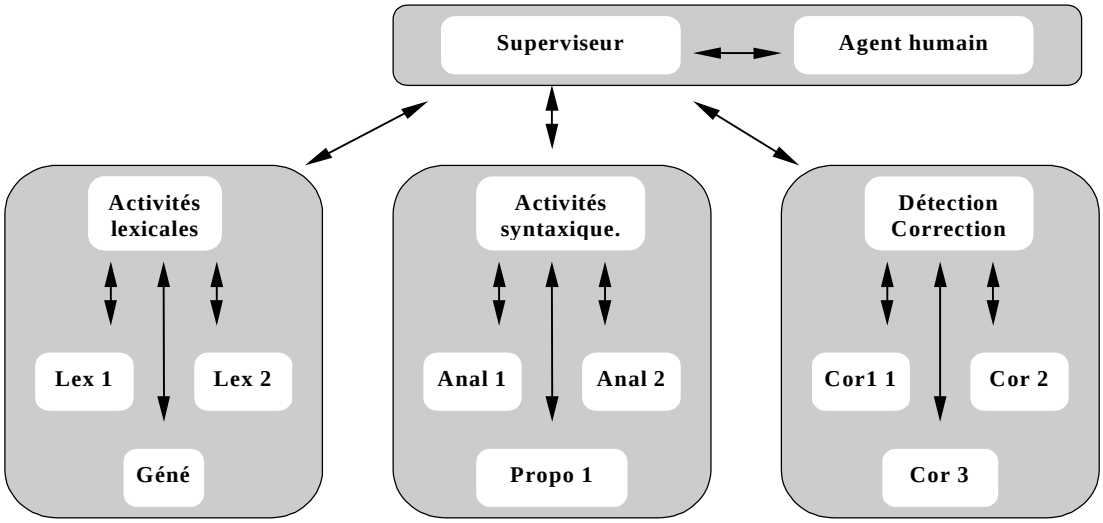


Figure 9.1 : Architecture générale

Sur le plan du contrôle, l'architecture est une architecture hiérarchique pseudo pyramidale hybride (figure 9.1). Nous allons justifier cette appellation.

9.2.1.1 Architecture hiérarchique pseudo pyramidale :

1. Un *superviseur* contrôle l'action de quatre *pilotes*. Cet agent superviseur va être dénommé *CELINE* car il va être un peu le maître d'œuvre de la correction des erreurs et sa principale fonction se situe au niveau *stratégique*. Résumons son rôle :
   - ⇒ Ce superviseur est le module du correcteur en liaison avec l'éditeur du traitement de texte.
   - ⇒ Un agent de travail dénommé *BRUNO*[9] va, sur ses indications, segmenter le texte en phrases et les phrases en mots.
   - ⇒ Le superviseur va décider de l'activation ou non d'agents de traitement de domaines particuliers ou d'agents de traitement des

[9] *BRUNO* n'est pas interprétable en tant que sigle. C'est le petit-frère de CELINE et à un an, il démonte tout !

exceptions en fonction de ses connaissances relatives au rédacteur. Cette connaissance et l'activation a priori de ces agents permettent une optimisation du coût de traitement mais à défaut, ces agents pourraient aussi être activés lors du fonctionnement normal du système (§ 9.2.2.2).

⇒ Le superviseur va activer par l'intermédiaire des pilotes, la vérification lexicale, la vérification syntaxique, la vérification des accords (accord des verbes, accord en nombre et genre …) et la mise à jour des statistiques.

⇒ Par ailleurs, il va aussi surveiller le travail se faisant dans le blackboard, en tenir informés les pilotes et décider de la fin de ce travail.

⇒ *CELINE* va activer le *MAEC* (chargé des mises à jour du MLPS du rédacteur)

⇒ *CELINE* va activer le *DSR* (chargé de la mise à jour des connaissances du système sur lui-même).

⇒ *CELINE* va correspondre avec l'agent humain.

2. Chaque pilote est responsable d'un *secteur d'activités.* et contrôle le travail de plusieurs *agents*. Nous verrons que la tâche d'un agent peut être considérée comme une *tâche logistique* de gestion des agents disponibles. Les quatre pilotes sont :
   ⇒ le pilote des activités lexicales : *PAL,*
   ⇒ le pilote des activités syntaxiques *PAS,*
   ⇒ le pilote des activités de correction des erreurs *PAVA,*
   ⇒ le pilote des activités statistiques *PAST.*

3. Chaque agent est spécialiste d'un domaine étroit mais peut dépendre de plusieurs pilotes.

4. Un agent particulier : l'agent humain communique avec le superviseur, les pilotes et certains agents à caractères cognitifs.

9.2.1.2 Architecture hybride

Cette architecture est aussi hybride car la communication va se faire sous deux formes différentes :

- communication par message pour les activations et pour quelques rares interactions directes,
- communication par blackboard pour le travail fondamental.

9.2.2 MODES DE COMMUNICATION

Une application répartie multi-agents peut se voir comme un ensemble de processus communicants; la communication comprenant un double aspect : des échanges d'informations et une synchronisation entre les processus.

On distingue plusieurs modèles de communication :

- Echange de messages entre processus (modèle dit « processus, messages, portes »). Ce sera notre choix pour les communications par messages.

- Clients-serveurs pour lesquels ils existent des outils (appel de procédures à distance). Nous ne retiendrons pas cette solution.
- Modèle de mémoire virtuelle partagée dans laquelle nous placerons notre choix d'un blackboard.

9.2.2.1 Communication par blackboard

Le protocole de la communication par blackboard des agents de travail à l'occasion de leurs travaux (intervention sur la structure de données du blackboard) a été progressivement introduit tout le long des chapitres et nous ne reviendrons pas sur ce point.

La communication par blackboard présente l'inconvénient de constituer un goulet dans la circulation des informations et contribue à un ralentissement du système.

La communication par blackboard semble s'imposer lorsqu'il s'agit de prendre une décision contextuelle. En effet, à une date t, le blackboard contient l'ensemble des informations disponibles à cet instant. On peut alors accepter de payer le coût de ce mode de communication.

Par contre, dans la recherche d'un allégement du flot d'informations circulant vers ou depuis le blackboard et lorsqu'il s'agit d'un travail ponctuel bien défini, il peut sembler avantageux de faire correspondre directement deux agents.

Par exemple supposons que l'agent DCFA détermine qu'un mot initialement au pluriel doit être mis au singulier. Pour prendre cette décision, l'agent DCFA a utilisé le contexte du mot et donc le blackboard. Pour la prise en compte de cette décision le protocole d'action, tel que présenté jusque là, va demander des accès au blackboard permettant :

1. de mettre une marque de validité de refus de la forme au pluriel,
2. de modifier toutes les structures se référant à l'ancienne forme,
3. de générer une nouvelle ligne avec la valeur *singulier* de la variable morphologique *nombre*,
4. d'attendre que le secteur lexical se soit aperçu de cette nouvelle ligne et ait généré la forme textuelle au singulier.
5. de créer toutes les structures se référant à la nouvelle forme.

Si nous supposons maintenant que l'agent DCFA s'adresse directement à l'agent lexical concerné pour lui demander la nouvelle forme textuelle, à l'étape 2 nous pouvons alors générer une ligne correcte avec la forme textuelle et la variable *nombre* correctes. Nous économisons au minimum l'étape 4 et, de plus, nous pouvons alors regrouper et optimiser les étapes 2 et 5 sur les structures.

Nous allons donc maintenant nous intéresser plus particulièrement à la communication par messages.

9.2.3 COMMUNICATION PAR MESSAGES

9.2.3.1 Problématique minimale

9.2.3.1.1 Modèle « processus, messages, portes »

Ce modèle repose sur deux procédures *envoyer* et *recevoir* :

Version primaire :

Envoyer (message, processus)	Primitive non bloquante
Recevoir (processus, message)	Primitive bloquante (attente du message)
Un problème : l'identification des processus. Sur un même site cela ne pose pas de problème (pid) mais un système réparti va demander une solution du type estampille ou ...	

Version plus évoluée : utilisation d'une porte, boîte à lettres dans laquelle on peut déposer et recevoir du courrier :

Envoyer (message, porte)	Primitive non bloquante en général (option : bloquante jusqu'à un accusé de bonne réception par la porte)
Recevoir (porte, message)	Primitive bloquante (attente du message)
Problème de désignation encore : Les identificateurs de portes sont des identificateurs uniques dans le système.	

Le développement sous UNIX permet de disposer d'un type de portes particulières les sockets.

9.2.3.1.2 Message d'activation et synchronisme

Pour les messages d'activation, il n'y a pas vraiment de synchronisme (figure 9.2). Un agent 1 activant un agent 2 pour une tâche va envoyer un message (voir au paragraphe 9.2.3.2,.le détail de la communication avec un socket d'écoute et des sockets de travail). L'agent 1 va poursuivre sa tâche. L'agent 2 de son côté va travailler sur le blackboard.

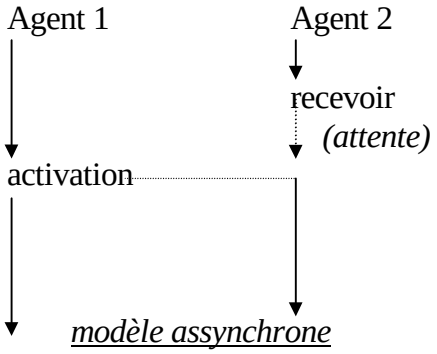


Figure 9.2 : Asynchronisme et activation

9.2.3.1.3 Traitement des requêtes et synchronisme

On peut envisager différents cas :

- **Demande à un agent monotâche.**

Schéma de fonctionnement d'un agent monotâche	
Agent i	Agent k (monotâche)
... envoyer (message,S) autres instructions recevoir (Pi, message) quand on a besoin ...	Début recevoir (S) traiter message executer service message_retour := résultat envoyer (Pi,message_retour) aller à debut fin

Ce mode de fonctionnement peut simuler un fonctionnement synchrone comparable à celui de l'appel de procédure à distance : on demande une certaine tâche et on attend la réponse (figure 9.3).

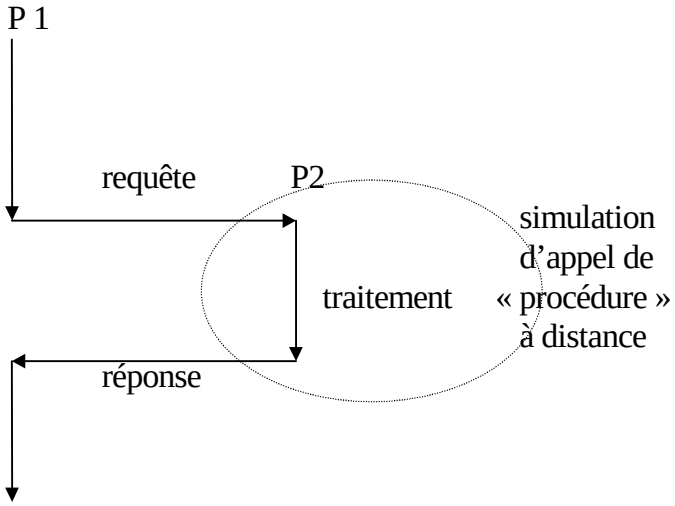


Figure 9.3 : Modèle synchrone de communication

On peut réaliser un modèle « message porte » en utilisant des boîtes à lettres (figure 9.4).

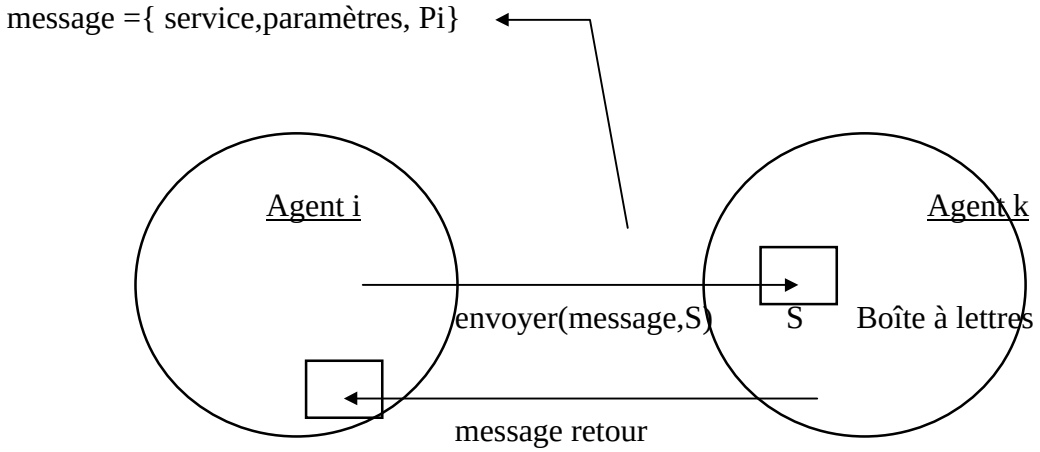


Figure 9.4 : Utilisation d'une boite à lettre

La boite à lettre de l'agent k va être une file d'attente de requêtes, file pouvant être gérée de diverses façons (classiquement FIFO).

- **Demande à un agent multi-tâches**

Un processus veilleur va attendre une requête et la sous-traiter.
Pour sous-traiter la requête deux solutions sont possibles :

- Un pool de processus de travail qui exécutent :
  attente d'une commande du processus veilleur,
  exécuter le service,
  envoyer la réponse.
  Inconvénient principal : nombre limité de processus dans le pool et donc possibilité d'attente.

- Création dynamique de processus pour exécuter un service
  On utilise des processus « légers » dénommés threads (un seul espace d'adressage pour plusieurs flots d'exécution). Sur une machine UNIX on obtient ainsi une simulation de parallélisme.

Processus veilleurs	Processus léger
attendre la requête créer processus léger	exécuter service message retour := résultat envoyer (message retour) kill

  Inconvénient principal : saturation du serveur.
  Malgré cet inconvénient nous avons retenu cette solution en reprenant notre hypothèse des performances suffisantes de machine.

9.2.3.2 Solution retenue : threads et sockets

L'agent de travail père va pouvoir être dupliqué en autant d'exemplaires fils (fonction UNIX fork) que nécessaires (figure 9.5).

Initialisation de la communication

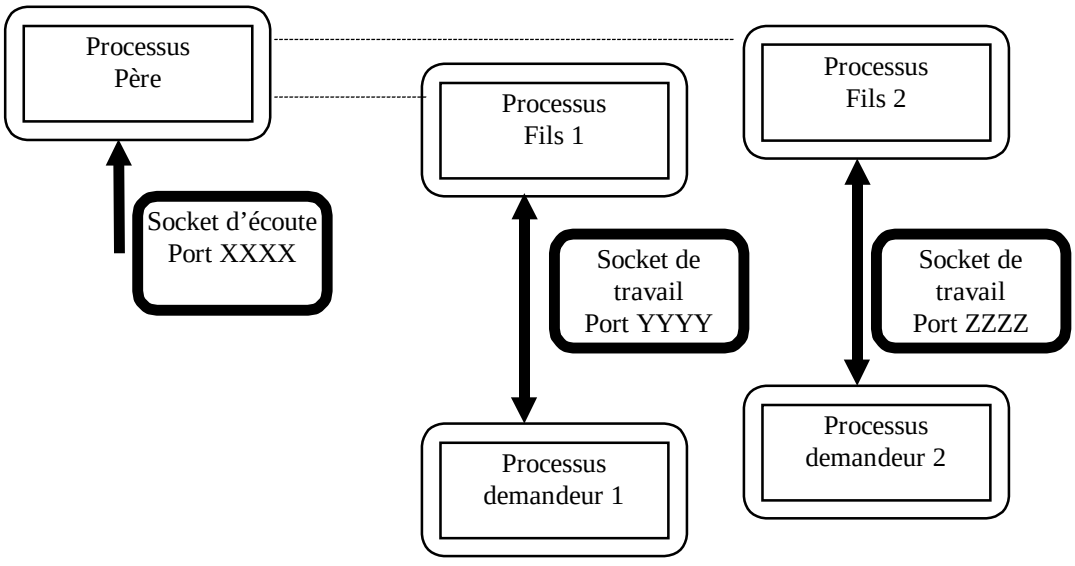


Figure 9.5

Le processus père va être en attente d'une communication sur un socket d'écoute (port désigné par son numéro).
L'agent demandeur d'un certain travail va appeler le serveur sur ce socket d'écoute. Cet appel nécessite de la part de l'agent demandeur la connaissance du site du serveur et du numéro du port du socket d'écoute du processus père.

Réception d'un appel
Le processeur père va créer un fils. Ce fils va communiquer, avec l'agent demandeur, sur un socket de travail différent du socket d'écoute. Le processus père va rester à l'écoute sur le socket d'écoute ce qui lui permettra de satisfaire la demande d'un nouveau client.

Travail d'un fils

- ***Validation de la demande :***
  Cette validation va comprendre deux étapes :
  1. Vérification de l'autorisation d'accès du processus demandeur et identification du demandeur. Cette reconnaissance va permettre si nécessaire d'effectuer des traitements sélectifs fonctions du demandeur (par exemple choix d'une matrice de transitions appropriée, sélection d'une table des couples mots à remplacer/mots remplaçants).
  2. Vérification de la conformité du message délivré vis-à-vis du protocole d'échange et identification du traitement demandé.
- ***Exécution du traitement demandé***
- ***Envoi de la réponse***

9.2.3.3 Structure des messages

Rappelons que la communication par message est principalement destinée à l'activation des agents :

- Activation des pilotes ou d'agents de travail par le superviseur
- Activation des agents de travail par les pilotes.

Quelques communications directes entre agents se font aussi par messages. L'objet de ces communications porte principalement sur l'activation directe d'agents et sur des demandes d'intervention ponctuelles.

Il n'existe pas de protocole général de communication entre agents mais un protocole par couple d'agents tenant compte de leurs compétences. Ces protocoles font partie de leurs accointances.

La structure des messages comprend une partie d'identification et une partie variable selon l'information passée ou demandée.

Le message ne contient pas d'identification de l'agent destinataire puisque, avant l'envoi du message à proprement parler, il y a eu contact sur son socquet d'écoute et que le message est envoyé sur le numéro du socquet de travail de l'agent.

Pour illustrer les variations possibles de messages nous allons considérer des agents du secteur d'activité statistique.

- Exemple 1 :

Message			
Entête d'identification			
Demandeur	Rédacteur	Tâche demandée	Corps
CELINE	*Einstein*	*BB*	*N*

Dans cet exemple *CELINE* demande à l'agent de désambiguïsation markovienne d'intervenir sur le blackboard et lui donne pour le choix des matrices de transitions l'identification du rédacteur *Einstein.* Le corps du message est *vide* (*Nill N*).

- Exemple 2 :

Message			
Entête d'identification			
Demandeur	Rédacteur	Tâche demandée	Corps
CELINE	*Einstein*	*P*	*0.450 0.500 det adji verb adv*

Dans cet exemple le pilote PAS des activitées syntaxiques demande à l'agent PSCM, pour l'utilisateur Einstein, une proposition de catégories morphologiques avec indication des seuils (0.450 seuil de probabilité pour l'ordre 1, 0.500 seuil de probabilité pour l'ordre 2) et indications du contexte (det adji pour le contexte gauche et verb adv pour le contexte droit).

- Exemple 3

Message			
Entête d'identification			
Demandeur	Rédacteur	Tâche demandée	Corps
MAEC	*Einstein*	*A*	*det+adji+subc+verb+adv*

Dans cet exemple l'agent MAEC demande à l'agent *PSCM*, de traiter, pour apprentissage des matrices de transitions du rédacteur *Einstein*, la chaîne *det + adji + subc + verb + adv*.

- Exemple 4 :

Message			
Entête d'identification			
Demandeur	Rédacteur	Tâche demandée	Corps
MAEC	*Einstein*	*A*	*N(ill)*

Dans cet exemple l'agent *MAEC* demande à l'agent *PSCM*, de réinitialiser à zéro les matrices de transitions du rédacteur *Einstein*.

9.2.4 CLASSIFICATION DES AGENTS DE TRAVAIL

Les agents (au sens générique) à mettre en œuvre peuvent être divisés en quatre classes différentes regroupées en deux niveaux : les agents de contrôle (le superviseur, les pilotes, l'agent humain) et les agents de travail dont nous allons préciser une classification.

9.2.4.1 Les agents de traitement direct des entrées sorties du système

Ces agents interviennent sur le texte en entrée et en sortie du traitement de texte. Ils demandent donc une interface ou au minimum un paramétrage dépendant du traitement de texte utilisé tel que par exemple les caractères d'exceptions pour le format.

- L'agent *BRUNO*. Cet agent va recevoir du superviseur *CELINE* la portion de texte à traiter dans le cas d'un fonctionnement en mode interactif ou bien le nom et le chemin directeur du fichier contenant le texte à traiter dans le cas d'un fonctionnement en mode automatique (voir § 5.3). Dans ce dernier cas, BRUNO devra interpréter l'extension du nom de fichier pour déterminer le type du traitement de texte.

- L'agent PONCT. Cet agent va vérifier les intervalles inter-mots et la ponctuation (voir § 5.3).

9.2.4.2 Les agents de traitement particulier spécifiques à un rédacteur

Les connaissances de ces agents pour un rédacteur particulier font partie du MLPS de ce rédacteur.

Dans ces agents interviennent :

- Les agent *MAEC* et *DSR* dont nous avons défini les fonctions au chapitre 4 sur les modèles humains.

- L'agent *TJRP*. Il correspond au "toujours remplacer" des correcteurs orthographiques du commerce. L'utilité de cet agent peut s'appliquer à différentes tâches :
    1. Cet agent mémorise les fautes morphologiques "habituelles" d'un utilisateur par exemple les fautes systématiques d'accents (*stratègie* toujours remplacé par *stratégie*).
    2. Cet agent peut servir aussi à des remplacements de sigles par des libellés complets.

- L'agent *CCFH* (Correction Contextuelle des Fautes Lexicales Habituelles). Le fonctionnement est équivalent à celui de l'agent TJRP mais le remplacement fait intervenir le contexte (par exemple *les taches de graisse* et *les tâches de l'Etat).*

9.2.4.3 Les agents de traitement des exceptions

Ces modules permettent de traiter les "exceptions" autres que celles rattachées à un rédacteur particulier qui sont traitées par les agents de la classe précédente.

Ces agents permettent d'obtenir un *lissage de la phrase* c'est-à-dire une phrase dans laquelle les exceptions sont remplaçables par du vocabulaire grammatical non terminal décoré par l'application de règles particulières (principalement incidence au niveau syntaxique).

> Exemple :
> L'ensemble *le XX^{e} siècle* peut être considéré comme un groupe nominal dans lequel le chiffre *XX* joue un rôle.

Dans cette catégorie pourraient intervenir des agents divers (Letellier 93) (Stefanini 93) permettant la reconnaissance de particularités telles que :

⇒ les abréviations,
⇒ des nombres ou des chiffres arabes …,
⇒ les dates et heures,
⇒ …

Les connaissances des agents de traitement des exceptions sont un ensemble de cas particuliers en général connus d'une manière extensive et portant sur du vocabulaire terminal du langage. Cette connaissance est en général peu volumineuse.

9.2.4.4 Les agents de traitement des domaines particuliers

Dans cette catégorie pourraient intervenir des agents divers (Letellier 93) (Stefanini 93) (Maurel 95), permettant la reconnaissance des particularités telle que :

⇒ les formules diverses (mathématiques, physiques ou chimiques …) et les langages de liaison correspondants,
⇒ reconnaissance de noms de lieux
⇒ reconnaissance de portions de texte traitant de domaines dont les expressions sont basées sur des codes langagiers échappant à la langue habituelle telle que : droit, bulletin météo, plan de vol d'un aéronef, ordres de combat d'un militaire. Cette liste quasi infinie demande pour chaque domaine un agent particulier spécialiste.
⇒ …

Ces agents sont susceptibles de donner tout à la fois les résultats des analyses lexicales, syntaxiques ainsi que la vérification des accords.

Dans leurs travaux de reconnaissance, ces agents appliquent des algorithmes complexes différents de ceux nécessaires à la simple reconnaissance d'une graphie. Nous n'envisagerons pas dans cette classe des agents traitant par exemple des noms propres car nous les rattacherons à des connaissances statiques contenues dans des lexiques.

Les langues naturelles étant riches en domaines, la présence de ces domaines particuliers rend inefficace l'application d'outils lourds portant sur l'ensemble de la langue. L'application préalable de ces agents permet de réduire le degré d'ambiguïtés et devient donc souhaitable.

Par contre le grand nombre potentiel de ces agents rend trop coûteuse la possibilité de les appliquer tous a priori. La stratégie va consister à ne les appliquer que si une information apporte une suspicion du bien fondé de leur utilisation. Cette information pourrait se trouver dans le MPLS du rédacteur. Cette information peut aussi s'obtenir après l'exploration inutile de beaucoup d'autres possibilités quand (enfin !) le système aura déterminé le bon agent. La première rencontre/reconnaissance va donc être très coûteuse mais une autre rencontre dans le même texte devrait alors être traitée rapidement.

Exemple : Les ordres d'un commandant d'escadrille de l'armée de l'air donnant par écrit ses instructions.

Le texte, outre un minimum de langage courant, va contenir du jargon météorologique, du jargon d'aviateur relatif au plan de vol et enfin du jargon militaire. Le rédacteur peut dans ce cas indiquer une bonne fois pour toute que, dans les documents de ce type, il y a présence des trois jargons.

La stratégie du traitement du texte va alors consister à faire la reconnaissance des parties du texte probablement majoritaires se rattachant à ces trois domaines et ensuite seulement à traiter le restant non reconnu du texte.

Le traitement d'un texte de ce type sans idée préconçue va amener une multitude d'essais négatifs relatifs à de nombreux domaines avant de tomber sur les bonnes relations *partie du texte* ⇔ *jargons utilisés*. Il importe après cette coûteuse détection de mémoriser pour ce texte cette possibilité. Nous retrouvons ici l'aspect dynamique de la stratégie qui doit savoir détecter les informations reçues et les prendre en compte.

9.2.4.5 Les agents généraux

Les connaissances de ces agents sont à l'échelle de l'ensemble d'une langue naturelle. Ils peuvent être divisés en :

- Agents de traitements généraux dont les connaissances sont, ou bien un ensemble de règles génériques utilisant presque exclusivement des symboles non terminaux de la grammaire, ou bien des connaissances presque exclusivement du domaine du vocabulaire terminal. Dans cette catégorie nous trouvons analyseurs et générateurs morphologiques (agent *PILAF*), analyseurs et générateurs phonétiques, analyseurs syntaxiques (agent *PILCHART*) ...

- Agents d'aide à la décision des agents de traitements généraux. Par exemple, pour un mot inconnu ou pour lever une ambiguïté morphologique : proposition de catégories morphologiques s'appuyant sur des modèles de Markov (agent *PSCM*) ou sur la Méthode des Cartes (agent *PILCHART).*

- Agents experts de la correction d'erreurs. Parmi ces agents nous trouvons *COMBI* (un agent proposant des formes textuelles par traitement alphabétique à partir des graphies non-reconnues) et *DCFA* l'agent de vérification-correction des fautes d'accord.

9.3 CELINE ET L'EXISTANT

9.3.1 PARAMÉTRAGE DE CELINE

Dans le souci de donner à CELINE une portée la plus générale possible, nous avons convenu de conserver le paramétrage défini à l'occasion de l'étude de corpus issus de l'oral (Frechet 93). En effet, ce paramétrage couvre les textes de l'écrit traditionnel et permet de faire face à l'inclusion d'une partie du type dialogue, discours …

Un autre intérêt est de disposer d'un corpus désambiguïsé lexicalement, ne comportant pas de fautes orthographiques, mais absolument épouvantable sur le plan de la construction syntaxique.

Nous distinguerons donc :

- 60 catégories grammaticales traditionnelles comme pronom personnel, verbe, substantif commun, adjectif qualificatif, ...
- 7 classes syntaxiques comme début de phrase, début de reprise, début d'incise, ...
- 22 catégories acoustiques comme claquement de langue, inspiration ou expiration buccale audible, inspiration ou expiration buccale bilabiale, inspiration ou expiration nasale audible,...

Les marqueurs syntaxiques et acoustiques étant des résultats d'expérimentation pouvant bruiter nos propres résultats, nous n'avons retenu pour les expérimentations que les soixante catégories grammaticales et un marqueur acoustique (*resp* respiration), soit un total de 61 catégories. Le traitement réel de l'oral demanderait d'utiliser des agents spécifiques du domaine.

Par ailleurs, nous ajouterons le concept de "super-classe", partition de l'ensemble des catégories morphologiques. Par exemple, la super classe *sverb* réunit les catégories *verb* (verbe), *xav* (auxiliaire avoir), *xal* (auxiliaire aller), etc.

9.3.2 CHOIX D'UN LANGAGE DE DÉVELOPPEMENT ET D'UNE INTERFACE

L'annexe B présente PILAF et l'environnement de travail. Nous n'expliciterons donc pas les éléments correspondants que nous supposerons connus.

L'objectif était (et est toujours d'ailleurs) de porter les applications et leurs interfaces sur un ensemble de trois plates-formes : Macintosh avec le système 7, PC avec Windows 95 et terminal X ou page HTML sur machine UNIX.

Dans le but de réaliser un ensemble logiciel le plus portable possible, l'équipe a choisi le langage LISP pour certains modules, le langage C ou le langage C++ pour d'autres.

Les modules existants en LISP (méthode des cartes, analyseur de TOMITA) ne se sont pas révélés portables.

Le langage C présente une réelle portabilité. PILAF a été complètement réécrit en langage C (Genthial 96) et tourne sur les trois plates-formes.
Il en est de même de tous les agents développés en C (par exemple : agents markoviens, détection et vérification des accords)

Ce n'est pas le cas du C++ qui, même sans faire intervenir les interfaces graphiques, demande souvent une adaptation au système de développement. Pour être clair une application développée sous C++ Microsoft ne tourne pas sous du Borland C++ ou du Code Warrior. Les adaptations semblent très coûteuses.

Dans le cas du C++, la recherche d'interfaces graphiques portables nous a fait perdre beaucoup de temps sans que nous arrivions à des résultats intéressants. Nous avons noté que parmi les noyaux dits portables du commerce ceux que nous avons essayés ne fonctionnaient pas et étaient bogués.

Concernant les interfaces graphiques, ma conclusion personnelle est, qu'à moins de se spécialiser dans ce domaine et en absence d'outils de portage, il est plus rapide de réécrire les applications complètement pour les trois plates-formes plutôt que de chercher des adaptations.

Notre choix final :

- Pour les agents de travail, transparents au rédacteur, il n'y a pas lieu de développer une interface graphique et le langage C semble le plus adapté à moins que la spécification du logiciel n'implique la programmation par objets pour laquelle le langage C++ s'impose.

- Pour les modules faisant intervenir une interface avec le rédacteur, il faut bien en passer par les interfaces graphiques. La décision a été de regrouper alors autant que faire se peut les divers modules avec des interfaces graphiques.

- Nous envisagons aussi une autre solution : un noyau en C ou en C++ et une interface avec un langage de script de type Visual Basic ou Tcl/Tk.

Résumons : en 1994 au début de la thèse, aucun module portable n'était disponible (PILAF version C a été porté UNIX en 95/96 et sur PC en 97). Les premiers prototypes des agents de CELINE, « testeur d'idées », ont donc été développés sur Mac où nous disposions de PILAF et de PILCHART. Plusieurs agents ont été alors écrits en C : les agents Markoviens, un agent de segmentation des analyses syntaxiques en structures et un agent de vérification-correction des fautes d'accord. La simulation du parallélisme et des interactions se faisait par écriture et lecture de messages par l'intermédiaire de fichiers.

9.3.3 CONCLUSION

Des modules préexistants nous souhaiterions utiliser après réécriture :

- PILAF comme lexique général.
- Les deux analyseurs PILCHART et TOMITA. L'utilisation de deux analyseurs théoriquement équivalents permet de les mettre en concurrence et de comparer leurs performances ainsi que leur adéquation avec le système général de CELINE. PILCHART se prête mieux par sa LAT à des analyses incrémentales et permet d'obtenir les structures (sous-arbres). Tomita permet d'intégrer une composante de parallélisme.
- Nous garderons en réserve CADET (grammaire de dépendances) dont l'utilisation pourrait être complémentaire de celles des analyseurs par grammaires de constituants. CADET présente l'avantage de pouvoir introduire un minimum de sémantique.

9.4 STRUCTURE DE DONNÉES D'UN CORPUS : AGENTS BRUNO ET FORME

9.4.1 STRUCTURE MINIMALE

Nous allons nous placer dans le cadre de la correction-détection des erreurs automatique. Le texte à traiter est lu par l'agent « BRUNO » à qui le superviseur passe en paramètres le nom du rédacteur et la désignation du fichier contenant le texte.

S'il s'agit d'un rédacteur connu, BRUNO va récupérer dans le MPLS de ce rédacteur, le tableau spécifique des séparateurs s'il en existe un ; dans le cas contraire il va choisir un paramétrage par défaut.

BRUNO va alors segmenter le texte en phrases et chaque phrase en mots. Pour permettre un parallélisme des traitements, le texte à traiter est mémorisé sous la forme de trois matrices (dont les indices sont le numéro de la phrase et le numéro du mot dans la phrase) :

- la matrice des mots,
- la matrice des formats,
- la matrice des séparateurs.

Une phrase se définit à l'aide de séparateurs (par exemple le point). Lorsque nous parlerons d'une phrase, nous utiliserons le terme de vecteur. Une phrase est décomposée en trois vecteurs : le vecteur des mots, le vecteur des formats mémorisant police, taille, attribut (gras, italique, ...) et le vecteur des intermots (blancs, ponctuation etc.). Il y a autant de mots que de séparateurs (une phrase débute sans séparateur et se termine par un séparateur).

> Par exemple le texte : *Il fait beau aujourd'hui. Je vais , heureux, me promener.*
> va correspondre à l'initialisation des matrices *mot* et *sep* suivantes :
>
mot(1,1) :	=Il=	mot(2,1) :	=Je=
sep(1.1) :	= =	sep(2,1) :	= =
mot(1,2) :	=fait=	mot(2,2) :	=vais=
sep(1,2) :	= =	sep(2,2) :	= , =
mot(1,3) :	=beau=	mot(2,3) :	=heureux=
sep(1,3) :	= =	sep(2,3) :	=, =
mot(1,4) :	=aujourd'hui	mot(2,4) :	=me=
		sep(2,4) :	= =
		mot(2,5) :	=promener=

L'état initial de la structure de données incluse dans la blackboard correspond à un remplissage par le vecteur des mots tel qu'il a été obtenu par l'agent BRUNO. Le travail de correction des erreurs va modifier les graphies, supprimer des mots par fusion, ajouter des mots par coupure, etc. Le travail fini, il faut bien restituer au rédacteur un texte rectifié au format.

Exemples :

> L'agent d'analyse morphologique va pouvoir regrouper deux mots en une seule locution en y incorporant le séparateur intermédiaire (" après-midi " initialement reconnu en deux mots " aprés " et " midi " et un séparateur " _ " va pouvoir être regroupé en un seul mot).

> Après détection d'une faute ('chavvaux) ou d'une faute d'accord (" les cheval boivent "), le pilote d'un module de génération morphologique va remplacer le mot " cheval " par " chevaux ").

Au total, onze primitives manipulent cette structure de données en trois vecteurs :

- insertion d'un mot
- insertion d'un format
- insertion d'un séparateur
- suppression d'un mot
- suppression d'un séparateur
- suppression d'un format
- modification d'un mot
- modification d'un séparateur
- modification d'un format
- fusion de deux phrases
- coupure d'une phrase en deux phrases.

Nous pouvons remarquer que les initialisations des vecteurs à partir du texte initial utilisent les trois primitives d'insertion alors que les modifications des vecteurs (à partir du blackboard) utilisent l'ensemble des procédures.

9.4.2 STRUCTURE COMPLÈTE

En vue de la correction automatique, nous ajouterons aux trois vecteurs précédents, le vecteur mémorisant l'ensemble des informations pour chacun des mots (lexique, catégorie morphologique, variables morphologiques etc.) décrit au paragraphe 6.1.1.3.8.

Rappelons que ces informations vont permettre, en cas de nouveau travail sur le même texte, de ne pas refaire l'ensemble des coûteuses recherches lexicales, syntaxiques ou de vérification des accords.

9.4.3 INTERVENTION DE L'AGENT FORME

Cet agent va traiter le vecteur des séparateurs et le rendre conforme au choix de présentation adopté. Beaucoup de systèmes de correction d'erreurs proposent un agent du même type ou des fonctionnalités équivalentes.

Cet agent va être activé par BRUNO à la fin de son découpage du corpus en trois structures. Ce travail (peu coûteux et rapide) va donc se faire en parallèle avec le début de traitement de la phrase.

La soudure ou la coupure de deux mots ou de deux phrases vont entraîner l'envoi de messages à l'agent BRUNO lui indiquant les mises à jour (ajout, suppression, modification) à appliquer dans les trois vecteurs structures de données.

En l'état du développement l'agent FORME se contente de vérifier et corriger les éléments :

1. Un seul intervalle entre deux mots.
2. Pas d'intervalle entre un mot et une virgule, un mot et un point, un seul intervalle après.

3. Un intervalle devant et derrière les signes doubles { ; ? !).
4. Un intervalle devant une parenthèse ouvrante et pas d'intervalle après. Un intervalle après une parenthèse fermante et pas d'intervalle avant.

9.5 LES AGENTS DE CONTRÔLE

9.5.1 LE SUPERVISEUR

Le superviseur va décider de la stratégie générale à mettre en œuvre et gère la planification des actions des pilotes et de certains agents.

Le choix du plan à mettre en œuvre dépend :

1) Initialement (c'est-à-dire au début de chaque nouvelle analyse) de paramètres statistiques dépendant facultativement d'informations sur le rédacteur ou sur le(s) domaine(s) traité(s). En l'absence d'informations sur l'utilisateur ou le domaine, les paramètres statistiques sont choisis par défaut.

2) Des *imprévus* amenant à modifier cette planification. Un imprévu correspond principalement au signal d'échec en provenance d'un des pilotes ou d'un des agents ou encore à une intervention de l'agent humain. La réception d'un avis d'imprévu déclenche l'application d'une règle heuristique de modification du plan.

> Exemples :
> 1 - L'agent humain peut interrompre le processus de correction d'erreurs. Le superviseur doit alors interrompre toutes les activités et envoyer des messages à tous les pilotes et à tous les agents qui dépendent directement de lui.
> 2 – Le problème est voisin dans l'hypothèse d'une panne soudaine de réseau ou de machine rendant certains agents indisponibles. L'adoption d'un nouveau plan dépend de la possibilité ou non de remplacer les sites indisponibles par d'autres équivalents.
> 3 - En cas de changement d'une valeur seuil ou d'une anomalie par rapport à une valeur seuil. Par exemple, il pourrait arriver qu'un rédacteur connu pour lequel on connaît les domaines et le MPLS change brutalement de domaine et de style (chercheur en électronique écrivant un poème …). Comme nous l'avons vu, par le suivi des coefficients, le système va détecter ce changement et le superviseur doit réagir en conséquence.
> 4 – Changement du taux d'auto-correction (4.2.5) entraînant un mode de correction différent et une nouvelle stratégie.

La classification des agents proposée nous conduit à un besoin de définition de règles heuristiques et à la recherche d'une stratégie dynamique dans l'ordonnancement des interventions des agents. En effet, en partant des classes d'agents de travail intervenant directement dans la correction-détection des erreurs, on peut opérer a priori de diverses façons. Parmi toutes les combinaisons possibles citons :

1. un ordonnancement séquentiel des classes,
2. un ordonnancement portant sur :
   - les agents de deux classes en parallèle,
   - les classes restantes,
3. tous les agents de toutes les classes en parallèle.

Par exemple, on peut imaginer, qu'en s'appuyant sur le suivi statistique et son jeu de coefficients, on décide d'appliquer d'abord :

- Les agents de traitements d'erreurs particulières si le taux d'erreurs dépasse un certain seuil et si la richesse linguistique (nombre de mots du PMLS) du rédacteur est faible. En effet peu de mots utilisés et un grand nombre de fautes veulent dire que cela va être toujours les mêmes fautes.
- Les agents de traitement des exceptions pour une personne faisant peu d'erreurs mais utilisant des tournures particulières, précieuses ou des expressions provenant du vieux français.
- Les agents des domaines particuliers pour le texte d'un rédacteur identifié comme utilisant ces domaines.
- Les agents généraux, pour un rédacteur complètement inconnu ou pour un utilisateur faisant très peu d'erreurs.

Autre type d'élément stratégique :

- Même en présence d'un grand nombre de processeurs, il semble inutile d'activer initialement des agents lexicaux pour lesquels les coefficients d'utilisation tombent en-dessous d'un seuil. En effet la gestion des communications et l'activité correspondante du processeur va ralentir le fonctionnement du système total. Il peut sembler plus astucieux, dans ce cas, d'attendre que les lexiques à fortes probabilités d'utilisation aient été explorés et de n'activer les agents à faibles probabilités que sur les mots non reconnus à l'étape précédente.

9.5.2 LES PILOTES

Les pilotes gèrent la planification des actions des agents qu'ils activent c'est-à-dire :

1) l'ordonnancement de l'intervention des agents de leur domaine,

2) éventuellement la portée de chaque intervention c'est-à-dire la partie de la sous-phrase à traiter. Par exemple, il n'y a pas lieu de re-traiter les parties d'un texte reconnues par un agent d'un domaine particulier. La prise en compte peut se réaliser de deux manières : pour les incrustations en milieu de phrase par une marque de validité particulière ANT (à ne pas traiter) avec affectation d'une catégorie particulière *cit* (citations).

Le choix d'un ordonnancement dépend des demandes du superviseur, des connaissances de l'agent et des données du blackboard.

Comme pour le superviseur, des imprévus peuvent amener à modifier cette planification. La réception d'un avis d'imprévu déclenche l'application d'une règle heuristique de modification du plan.

> Exemples :
> 1 - Le pilote *PAL* peut recevoir du superviseur l'information de passage au mode de correction automatique. Jusqu'à cet ordre, le pilote activait les agents lexicaux les plus probables et ne lançait pas des recherches sur l'ensemble des agents lexicaux disponibles en préférant demander l'aide du rédacteur. Dans le mode automatique au contraire, face à une forme textuelle inconnue, l'ensemble des lexiques va pouvoir être utilisé.
> 2 - Le pilote *PAS* peut recevoir du pilote *PAST* l'information que le corpus peut être désambiguïsé par l'agent MARKOV. Ce pilote doit alors arrêter les agents de son

> secteur, activer l'agent Markov puis relancer le travail lorsque l'agent lui communique la fin de son travail.

9.5.3 LE PILOTE PAL DE L'ACTIVITÉ "LEXIQUE"

Nous détaillerons le fonctionnement du pilote PAL en tant que modèle du fonctionnement des pilotes.

9.5.3.1 Rappels :

- Avant de faire appel au pilote PAL, le superviseur va chercher à réduire le potentiel d'ambiguïté en faisant appel à certains agents :
  1) Dans le cas d'un utilisateur faisant beaucoup de fautes avec, en particulier, un grand nombre de fautes se répétant, l'application préalable des agents *TJRP* et *CCFL* permet de diminuer le nombre de formes textuelles non reconnues.
  2) Si les connaissances sur le rédacteur le permettent : application des agents des exceptions et des agents des domaines particuliers.

- Rappelons que dans les chapitres sur le maître d'école et les modèles humains, nous avons déjà traité des domaines d'un utilisateur. Nous partirons donc de ce point et nous supposerons que nous disposons des divers coefficients déjà rencontrés : coefficient d'efficacité, coefficient d'utilisation, coefficient de crédibilité. Si le rédacteur est connu, les valeurs vont être les valeurs du PMLS, sinon les valeurs seront des valeurs par défaut.

9.5.3.2 Activation des agents lexicaux

Le nombre d'agents lexicaux activés dépend du nombre de processeurs et des ordres de grandeur des coefficients :

- S'il dispose de plus de processeurs que d'agents à activer : activation de tous les agents lexicaux en parallèle.
- Si ce n'est pas le cas (et nous nous sommes volontairement placés dans ce cas) le pilote va rendre actif le nombre d'agents possibles en les choisisant dans des files d'attentes d'activation. L'ordre de placement des agents lexicaux dépend des coefficients et de la stratégie choisie.

Le pilote va planifier l'activation des agents lexicaux en faisant intervenir une stratégie basée sur trois listes ordonnées dans l'ordre décroissant des coefficients d'utilisation :

1. Une liste n° 1 comprenant au maximum les 5 derniers lexiques utilisés (autre que le lexique général PILAF et le lexique du MLPS qui sont toujours élus). Cette liste va être construite en utilisant les divers coefficients présentés au paragraphe 6.1.1.
2. Une liste n° 2 non bornée contenant les agents lexicaux déjà intervenus dans le même texte mais différents des agents de la liste 1.
3. Une liste n° 3 d'activation des agents lexicaux restants, construite à partir de la liste générale des agents lexicaux possibles. Nous proposons dans le paragraphe suivant une méthode de recherche pour faire face à un nombre important de lexiques.

Ces listes sont gérées dynamiquement et mises à jour en continu (rôles des agents *MAEC* et *MLPS* au paragraphe 4.1.5).

9.5.3.3 Solution future : liste d'accès par lexico-thésaurus

9.5.3.3.1 Principe de la méthode

Nous proposons pour le futur, en présence de plusieurs dizaines (centaines ?) d'agents lexicaux un ordonnancement des recherches basé sur le système des thésaurus des bibliothèques qui ont exactement le même problème : la question *Comment identifier et trouver un ouvrage d'un domaine donné ?* remplaçant notre *question Comment face à un mot inconnu, trouver le bon lexique ?*. La recherche au hasard du bon lexique semble trop coûteuse voire même complètement farfelue si on envisage des milliers de lexiques. Elle reviendrait à chercher un ouvrage au hasard parmi des milliers d'autres en vrac.

Il faudrait donc que, dans notre *« lexico-thésaurus »*, les indications de l'ouvrage soit remplacés par l'identification du lexique (nom, numéro IP d'accés Internet, numéro du socket d'écoute…).

Les thésaurus sont basés sur le système des mots clés. La constitution d'une liste des lexiques les plus vraisemblables s'établirait à partir d'une étude, sur l'ensemble du texte, de la coïncidence des mots avec un mot-clé du thésaurus.

Les différentes étapes seraient alors les suivantes :

- Rechercher chaque forme textuelle dans PILAF et le lexique du MLPS (on ne considérera pas les agents des listes 1 et 2 du paragraphe précédent, listes vides en supposant un nouveau texte).
    - ♦ En cas de réussite, nous mémorisons (comme proposé au § 6.1.1.3.8) la référence du lexique dans le but évident de ne pas refaire plusieurs fois la recherche correspondante.
    - ♦ En cas d'échec :

      Considérer la forme textuelle examinée comme un mot clé potentiel et parcourir le thésaurus pour rechercher si ce mot clé est bien présent :

        - ⇒ Si échec : on ne tire pas de conclusion.
        - ⇒ Si la forme textuelle est reconnue comme mot clé, alors dans, l'arborescence du thésaurus, tous les lexiques des nœuds fils vont être des lexiques candidats où il y aurait lieu de faire une recherche de cette forme textuelle. On ne fait surtout pas la recherche de cette forme textuelle dans l'ensemble des lexiques ainsi trouvé. On établit un compteur par lexique ainsi déterminé. Ce compteur va être incrémenté à chaque nouvelle apparition du lexique correspondant.
- Lorsque ce travail est fait pour l'ensemble du texte, les scores des compteurs vont indiquer les lexiques les plus vraisemblables. Nous pouvons alors à partir de ces scores constituer notre liste 2 initiale.
- A partir de la liste et en respectant l'ordre des lexiques, l'identification de chaque forme textuelle va pouvoir alors commencer. En parallèle la liste 1 va se construire.

Les avantages de cette méthode sont importants :

- La méthode ne demande pas un investissement énorme (les thésaurus étant déjà informatisés, il est sans doute facile de récupérer leurs arborescences et de saisir les références des lexiques).
- L'énorme avantage de cette méthode est de ne pas demander une connaissance préalable du rédacteur. Mieux même, cette méthode pourrait constituer un moyen rapide de détermination des caractéristiques du rédacteur.

Un des inconvénients de la méthode est qu'elle demande que l'utilisateur ne fasse pas trop d'erreurs sous peine d'un nombre insuffisant voire nul de mots clés reconnus.

Si on considère le traitement d'un texte complet dans le cas d'un rédacteur inconnu, on peut imaginer que, pendant que le système essaye laborieusement de traiter la première phrase ce qui occupe déjà largement le superviseur, le pilote PAL effectue ce travail d'exploration lui permettant de construire la liste 2 : liste qui va permettre d'optimiser toutes les autres recherches.

9.5.3.3.2 Expérimentation en cours

Nous avons imaginé une simulation du lexico-thésaurus. Cette expérimentation a été menée conjointement avec un projet d'option informatique en classe de première et terminale S d'un lycée au troisième trimestre 97-98 et sera poursuivie durant l'année scolaire 98-99.
Nous avons effectué cette expérimentation en prenant le cas particulier des noms propres de grands savants du domaine des sciences physiques. Soulignons que notre objectif n'était pas d'identifier un nom propre ; auquel cas un seul lexique aurait suffit avec par exemple une recherche dichotomique.

Objectif :
Notre objectif était de tester l'optimisation des accès lexicaux par détermination d'association lexiques(s)-texte en se servant des mots clés du thésaurus.

Réalisation

1) Nous avons pris le niveau 1 d'un thésaurus et obtenu 33 mots clés en physique et 11 mots clés en chimie. Pour quelques uns de ces mots clés nous avons pris aussi le niveau fils soit 36 nouveaux mots clés.
2) Pour chaque mot clé nous avons créé un lexique rempli avec des noms propres du domaine.
3) Nous avons alors constitué un corpus d'étude à partir de dix textes scientifiques de vulgarisation ou d'approfondissement des manuels scolaires. Les textes étaient choisis par leur appartenance à un domaine précis. Le corpus a été réalisé soit par saisie directe soit par numérisation par scanner et reconnaissance de caractères.
4) Nous avons appliqué notre technique de comptage par lexique avec une recherche des mots clés par simple parcours linéaire de la table

Les résultats
Le nombre maximum de lexiques déterminés pour des textes d'une page environ a été de un à cinq.
Par contre le lexique dans lequel se trouvaient les noms propres des savants du domaine a été toujours classée en premier. Pour chaque nom propre, la probabilité d'accéder directement au bon lexique est donc de 1.

Cette probabilité est à rapprocher de celle que donnerait une recherche au hasard : 1/80 soit 0,0125.

Conclusion et extension
Nous considérons ce résultat comme validant notre hypothèse.
Nous comptons étendre ce type d'expérimentation au langage technique des sciences physiques.
Des premiers essais manuels montrent qu'à partir de 200 mots les lexiques utiles sont déterminés.

9.5.3.4 Contrôle au niveau du blackboard

9.5.3.4.1 Contrôle d'un agent

Lorsqu'un nouvel agent est activé, une *marque de présence* portant son identifiant est ajoutée dans toutes les lignes du blackboard. L'agent va ensuite modifier cette marque en marque de *validité* en fonction de son travail comme nous l'avons signalé au § 2.3.5.2. L'observation de l'ensemble des marques permet de faire un bilan sur le travail réalisé. Le superviseur pour l'ensemble des informations, un agent pour son propre travail et éventuellement pour le travail d'agents d'aide vont pouvoir effectuer ce bilan.

Pour en revenir au secteur lexical, rappelons qu'une fois que les agents lexicaux sont activés, ils vont travailler d'une façon indépendante et vont scruter le blackboard en examinant toutes les formes textuelles sur lesquelles ils n'ont pas encore apposé leurs marques de validité. Le superviseur, le pilote PAL et l'agent lui-même peuvent donc par l'intermédiaire des marques de validité savoir si l'agent a donné son avis sur toutes les formes textuelles proposées.

Deux questions se posent :

1. Lorsqu'un agent a donné son avis sur toutes les formes textuelles présentes dans le blackboard deux stratégies sont possibles :
   - Ou bien l'agent repasse en position d'agent éligible et libère le processeur.
   - Ou bien l'agent continue de scruter le blackboard. En effet celui-ci est en perpétuelle évolution et, à tout moment, une ligne peut être ajoutée, ligne que l'agent est susceptible de résoudre.
2. Qui prend la décision ? Le superviseur, le pilote PAL ou l'agent lui-même ?

Notre réponse à la première question :

- Si l'agent a su reconnaître au moins une forme textuelle, il reste actif. Sinon il retourne dans la liste des lexiques éligibles. Sa place dans la liste des agents lexicaux éligibles va dépendre d'un nouveau calcul des coefficients.

Notre réponse à la deuxième question sera plus difficile car plusieurs solutions ont leurs avantages et leurs inconvénients que nous allons passer en revue.

- Premier cas : le superviseur. Le superviseur va scruter le blackboard et peut donc à chaque balayage faire le travail correspondant. Inconvénients : il devra envoyer un message à l'agent et au pilote PAL pour les prévenir. De plus, en opérant de même pour tous les secteurs, on surcharge en tâches le superviseur

ce qui peut-être une cause de ralentissement du système. On alourdit aussi les connaissances du superviseur et en cas d'ajout d'un agent lexical, on doit modifier les accointances du superviseur et du pilote PAL.

- Deuxième cas : variante de la solution précédente : le superviseur envoie un message au pilote PAL qui le répercute sur l'agent concerné. Avantages : le pilote PAL n'est pas obligé de scruter le blackboard. Cette solution ne demande pas que le superviseur possède la connaissance des agents lexicaux. Le coût d'envoi des messages est équivalent à celui de la solution précédente.

- Troisième cas : le pilote PAL. Ce pilote doit alors scruter le blackboard pour déterminer où en sont chacun des agents lexicaux. Inconvénient : un agent de plus scrute le blackboard d'où un ralentissement. Avantage : moins de messages avec un seul message à l'agent. Autre inconvénient : dans la première solution, le superviseur avait tous les éléments pour se rendre compte de l'avancement de la tâche et de sa fin. Dans ce troisième cas, le pilote PAL devra informer le superviseur de la fin des activités lexicales.

- Quatrième cas : l'agent lui-même. L'agent doit, de toute façon, scruter le tableau. Avantage : un seul message au pilote PAL. Mais comme dans le cas précédent le pilote PAL devra informer le superviseur de la fin des activités lexicales. Nous avons rejeté cette solution plus coûteuse sur le plan du fonctionnement du système : les pilotes et le superviseur sont sur le site du blackboard alors que les agents peuvent être sur des sites lointains. Nous avons aussi préféré regrouper les aspects cognitifs sur les pilotes et sur le superviseur.

Nous avons donc retenu le deuxième cas, c'est-à-dire contrôle du superviseur avec envoi de messages au pilote PAL. Nous continuerons ce chapitre compte tenu de ce choix et nous appliquerons aussi ce mode de fonctionnement aux autres secteurs d'activités.

9.5.3.4.2 Contrôle/gestion de l'ensemble des agents lexicaux

Lorsque tous les agents élus ont donné leurs avis deux cas se présentent : ou bien toutes les formes textuelles ont été reconnues, ou bien ce n'est pas le cas.

- Si toutes les formes textuelles ont été reconnues, le pilote PAL va attendre de recevoir un message du superviseur l'avertissant de la réussite de l'analyse syntaxique. A la réception de ce message, tous les agents du secteur lexical, pilote compris vont être désactivés. En cas d'échec de l'analyse syntaxique, le pilote PAL va mettre en action des plans portant sur la coupure ou la soudure de phrases. Sans entrer dans les détails, le plan pour la coupure d'une phrase en deux sous phrases va être basé sur la présence d'au moins un arbre complet[10] pour une partie de la phrase traitée. Le plan sur la soudure de deux phrases va être déclenché par un échec de l'analyse syntaxique de deux phrases adjacentes (aucun arbre complet). L'action consiste alors à supprimer provisoirement le séparateur entre les deux phrases et à poursuivre le travail à partir du nouvel ensemble ainsi constitué. Nous noterons que le blackboard doit pouvoir

[10] En partant de l'axiome de la grammaire.

mémoriser la phrase en cours de traitement et au minimum la phrase précédente pour pouvoir faire face à ce type de plan.

- Si toutes les formes textuelles n'ont pas été reconnues, sur message du superviseur, le pilote *PAL* va activer les agents susceptibles de proposer des formes textuelles remplaçantes :
  - ⇒ Tout d'abord ceux utilisant des connaissances sur le rédacteur (par exemple clé phonétique ou alphagramme pour le *MPLS*) susceptible de diminuer le coût de recherche.
  - ⇒ Ensuite, les agents faisant intervenir les générations morphologiques (exemple *PILAF*).
  - ⇒ En dernier lieu l'agent *COMBI* de traitement alphabétique qui lui va devoir proposer, pour validation, ses productions aux différents agents lexicaux (§ II.2.3.3). En cas de validation, la forme concurrente est placée dans le blackboard.

9.5.3.5 Etat du développement du pilote PAL

Le pilote est convenablement développé et possède les fonctionnalités que suppose le fonctionnement que nous venons de décrire (exception faite de la détermination de la liste 2 avec un lexico-thésaurus).

9.5.4 LE PILOTE PAS

9.5.4.1 Tâche de contrôle du pilote PAS

Le rôle essentiel du pilote *PAS* va consister à activer les différents agents de son domaine, principalement analyseurs syntaxiques et agents de Markov, dans le but d'obtenir si possible au minimum un arbre syntaxique complet pour chaque phrase. Dans le cas de solutions multiples (plusieurs arbres complets) le pilote *PAL* va chercher à les classer par ordre décroissant de probabilité.

La problématique du contrôle est la même que celle rencontrée pour le pilote *PAL*. Nous ne la développerons pas, mais nous admettrons le même mode de fonctionnement que celui adopté pour le secteur des activités lexicales : le superviseur scrute le blackboard, informe le pilote *PAS* qui, à son tour, informe ses agents. Les agents interviennent directement sur le blackboard.

Le pilote *PAS* va aussi intervenir dans le cadre de la collaboration entre agents pour l'aide à la décision des autres pilotes (essentiellement pour fournir une catégorie morphologique ou bien pour faire affecter des coefficients à des solutions concurrentes)

Le pilote *PAS* va se trouver confronté à deux problèmes essentiels :

1. Des solutions complètes concurrentes (sous forme d'arbres syntaxiques complets), solutions qui vont demander une désambiguïsation.
2. Une ou des solutions incomplètes sous forme d'une forêt de sous-arbres pour une phrase mal construite, pour un mot manquant ou parce que la grammaire incomplète des analyseurs ne permet pas une solution satisfaisante.

Le pilote *PAS* et les agents du secteur syntaxique vont pouvoir chercher de l'aide auprès des autres pilotes ou agents. Par exemple, si on envisage un utilisateur faisant peu de

faute d'accord, l'agent de vérification des accords va peut être permettre de déterminer parmi les solutions concurrentes celle(s) qui n'entraîne(nt) pas de fautes d'accord.

Considérons un exemple :

- Le *PAS* peut recevoir une demande de proposition de catégories morphologiques pour un mot.
- Le *PAS* va alors s'adresser aux agents (*PSCM* et *PILCHART*) pouvant lui proposer une catégorie morphologique.
- Ces agents, grâce au blackboard géré par le superviseur, connaissent le contexte du mot. Ils vont rechercher les solutions (si elles existent) et calculer les probabilités « locales » de ces solutions. Une sélection basée sur le principe des seuils va alors permettre de proposer en retour la ou les solution(s) que l'agent juge la ou les plus pertinente(s), solution(s) affectée(s) d'un coefficient de probabilité.
- Le *PAS* affecte à chaque agent un coefficient de crédibilité. En cas de solutions concurrentes, ce coefficient va être appliqué aux probabilités « locales » des solutions proposées par les agents. Le *PAS* obtient alors un classement des diverses solutions affectées cette fois d'une probabilité « générale ».
- Nous plaçant dans le cadre d'une correction le plus possible automatique, la stratégie choisie est de retenir uniquement la (ou les) solution(s) affectée(s) de la probabilité générale maximale. La ou les solutions retenues par le *PAS* vont alors être transmises au superviseur.

Remarques :

1. Les gestions des différents seuils de décision et des coefficients de crédibilité sont dynamiques et diverses heuristiques sont applicables.
2. Les produits probabilité « locale » par coefficient de crédibilité conduisent à une dispersion des valeurs d'où une sélectivité plus facile.
3. En présence de propositions multiples du PAS, le superviseur va se trouver devant un choix :
   - s'adresser à d'autres pilotes ou agents pour obtenir une levée d'ambiguïté,
   - en conserver une au hasard (!),
   - formuler une demande de renseignement à l'agent humain en mode interactif.

9.5.4.2 Etat du développement du pilote PAS

Le développement du pilote n'a pas été jugé prioritaire en l'absence provisoire de plusieurs agents d'analyse syntaxique. En particulier, les coefficients d'efficacité ne sont pas vraiment pris en compte (valeurs prises égales à un).

Le pilote sait communiquer par message avec le superviseur et les agents de son secteur. Son fonctionnement est actuellement sommaire :

- Il est activé par le superviseur qui lui passe le caractère anonyme ou au contraire l'identité du rédacteur.
- Dans le cas d'un rédacteur connu, il commence par activer l'agent syntaxique du MLPS (*ASMLPS*) conformément au § 6.3) qui va travailler en agent autonome directement sur le blackboard.

- Il attend alors la réaction du superviseur. Celui-ci, par le biais des marques de validité, va déterminer si l'agent *ASMLPS* a su réaliser ou non l'analyse syntaxique complète de la phrase.
- Dans le cas d'un rédacteur inconnu ou en cas d'échec de l'agent *ASMLPS* (et message en conséquence du superviseur), le pilote *PAS* active l'analyseur syntaxique basé sur la méthode des cartes (*PILCHART*) et l'analyseur de *TOMITA* (en réalité des simulations).

9.5.5 LE PILOTE PAVA

9.5.5.1 Tâche de contrôle du pilote PAVA

Le rôle de cet agent est d'activer les agents de correction des fautes d'accord. Rappelons que ces agents sont potentiellement nombreux et correspondent au découpage des connaissances à travers les particularités de la langue avec la réalisation de modules (transformables en agents par leur interface) d'un domaine précis.

Comme pour les précédents secteurs d'activité, nous retrouvons le même mode de fonctionnement : le superviseur scrute le blackboard, informe le pilote PAVA qui, à son tour, informe ses agents. Les agents interviennent directement sur le blackboard.

Nous avons à notre disposition pour l'instant un agent unique DCFA qui vérifie les fautes d'accord par la méthode des structures que nous avons détaillée au chapitre 7. Rappelons que cet agent intervient sur la blackboard, selon les principes développés aux § 2.3.5 et § 4.1.1, en changeant des variables morphologiques d'une forme textuelle reconnue ou en éliminant, parmi les formes textuelles proposées en remplacement d'une forme textuelle inconnue, celles ne vérifiant pas les accords dans les structures proposées. Cet agent vérifie les accords sur l'ensemble de la phrase et traite par exemple les accords du groupe nominal et du groupe verbal. Nous travaillons à l'éclatement de cet agent « généraliste » en plusieurs agents spécialistes chacun d'un type de structure (groupe nominal, groupe verbal, conjonction…).

9.5.5.2 Etat du développement du pilote PAVA

Compte tenu de la pénurie en agents spécialistes, le rôle du pilote *PAVA* est très réduit :

- Il est activé par le superviseur.
- Il active l'agent DCFA.
- Lorsque le superviseur juge son rôle terminé, il désactive l'agent *DCFA* et se désactive lui-même.

9.5.6 LE PILOTE DES ACTIVITÉS STATISTIQUES

9.5.6.1 Tâche de contrôle du pilote PAST

Le rôle de ce pilote est d'activer les agents effectuant un travail de statistique à travers le blackboard. Nous avons vu les rôles de ces principaux agents :

- L'agent maître d'école *MAEC* (§ IV.1),
- L'agent *DSR* (§ IV.2),
- L'agent PSCM de proposition de catégories morphologiques,

- L'agent *PILMARK* de désambiguïsation morphologique dont nous avons étudié l'algorithme d'intervention au § V.5. Ce paragraphe nous a montré également différentes stratégies et une commutation dynamique dans le choix des stratégies.

La tâche de contrôle du pilote des activités statistiques reste simple, car, même dans l'hypothèse d'un système complètement développé, ce pilote n'a pas à mettre en concurrence des agents.
Une différence fondamentale avec les autres secteurs d'activité est que les travaux des agents du secteur ne se voient pas forcément au niveau du blackboard. Le superviseur scruteur des marques de validité ne dispose donc pas des moyens de vérifier la fin du travail de l'ensemble des agents.

Le protocole adopté est le suivant :

- En fin de tâche, chaque agent va signaler la fin de travail au pilote.
- Lorsque cela sera le cas, le pilote ira à son tour signaler au superviseur la fin de l'ensemble des travaux du secteur d'activités statistiques.

9.5.6.2 Etat du développement du pilote PAST

Le développement actuel du pilote PAST permet à celui-ci de remplir les tâches d'activation spécifiées.
Ce pilote doit être perfectionné sur le plan cognitif par une prise en compte plus détaillée du rédacteur et du contexte informatique (écrit saisi au traitement de texte ; oral, reconnaissance optique etc.).
Cette évolution doit également se répercuter sur le développement des agents (par exemple : introduction d'une stratégie de filtrage markovien d'ordre un utilisant l'algorithme de Viterbi etc).

9.5.7 CONCLUSION SUR LES PILOTES

Le système CELINE pourrait se passer des pilotes en transférant les tâches correspondantes sur le superviseur. La crainte est une surcharge de travail avec création d'un goulet. Le maintien des pilotes permet de ce fait de distribuer cette activité qui va pouvoir se faire en parallèle (parallélisme réel ou simulé selon le type de machine).

Sur un autre plan, le blackboard est un aussi un goulet dans la circulation d'informations. Le prototype actuel effectue une correction phrase par phrase. Une optimisation du système justifiant le maintien des pilotes pourrait être la suivante : pendant que les agents travaillent sur une phrase, le superviseur ou l'agent de pré-traitement *BRUNO* passe au pilote PAL, les mots de la phrase suivante. Le pilote PAL lance alors l'identification de ces mots. Pour préparer cette évolution, nous avons choisi une duplication des agents.

Le point de vue précédent se trouve renforcé si on admet que le pilote PAL doit effectuer un typage du texte en utilisant la recherche par mots clés et lexico-thésaurus.

En attendant un système avec plusieurs centaines d'agents lexicaux et compte tenu de son développement actuel, le système pourrait perdre son caractère hybride et évoluer à travers une suppression des pilotes vers une communication uniquement basée sur le blackboard.

9.6 COMPLÉMENTS SUR LES AGENTS

Nous n'allons pas reprendre le fonctionnement des agents. Nous les avons trouvés en temps utile dans notre démarche. Il peut toutefois être intéressant de présenter un récapitulatif de ces agents.

Notre souhait est d'utiliser un maximum d'agents lexicaux. Ces agents ne sont pas spécialement destinés au système CELINE, leurs intégrations demandent, outre les problèmes d'interface déjà invoqués, de préciser leurs fonctions. Nous allons donner une classification complémentaire de ces agents.

9.6.1 TABLEAU RÉCAPITULATIF DES AGENTS

Nom	Fonction principale	Paragraphes de référence
CELINE	Superviseur contrôleur de la bonne marche du système. Fonctions multiples sous jacentes dans l'ensemble de la thèse et principalement au chapitre IX	Thése ! Chapitre 9 ! 9.2.1 9.5.1
PAL	Pilote des activités lexicales	9.5.3
PAS	Pilote des activités syntaxiques	9.5.4
PAVA	Pilote des activités de vérification des accords	9.5.5
PAST	Pilote des activités statistiques	9.5.6
ALMPLS	Agent lexical du modèle linguistique partiel suffisant. Proposition par clé alpha	6.1
ASMPLS	Analyseur syntaxique du modèle linguistique partiel suffisant. Génération de structures	6.3
BRUNO	Pré-traitement du texte. Interface bidirectionelle entre le traitement de texte et le blackboard	9.4
CCFL	Correction contextuelle des fautes lexicales habituelles	9.2.4.2
COMBI	Recherche de formes textuelles remplaçantes par suppression, ajout et permutation de lettres	2.3.3
DCFA	Correction des fautes d'accord	7.3 Annexe D
DSR	Spécification du système par auto-apprentissage	4.2 4.3
FORME	Mise au format du texte (intervalles intermots, intervalles entre ponctuation et mots, majuscules etc.)	9.4.3
MAEC	Maître d'école Spécification du rédacteur par auto-apprentissage	4.1 4.3
PILAF	Analyseur et générateur morpho-syntaxique	4.6.2 Annexe B
PILCHART	Analyseur syntaxique basé sur la méthode des cartes Génération des structures	2.1.5
PILGRAM [11] *(GENREG)*	Filtrage grammatical correctif markovien Génération de règles de correction des modèles de Markov d'un rédacteur nouveau à matrices de transition non stable.	6.4
PILMARK	Désambiguïsation markovienne	5.5
PSCM	Proposition statistique de catégories morphologiques Construction des matrices de transitions des modèles de Markov	5.5
TJRP	Remplacement systématique	9.2.4.2

[11] Les deux agents GENREG et PILGRAM seront fondus en un seul.

9.6.2 SCÉMA RÉCAPITULATIF DU SYSTÈME CELINE

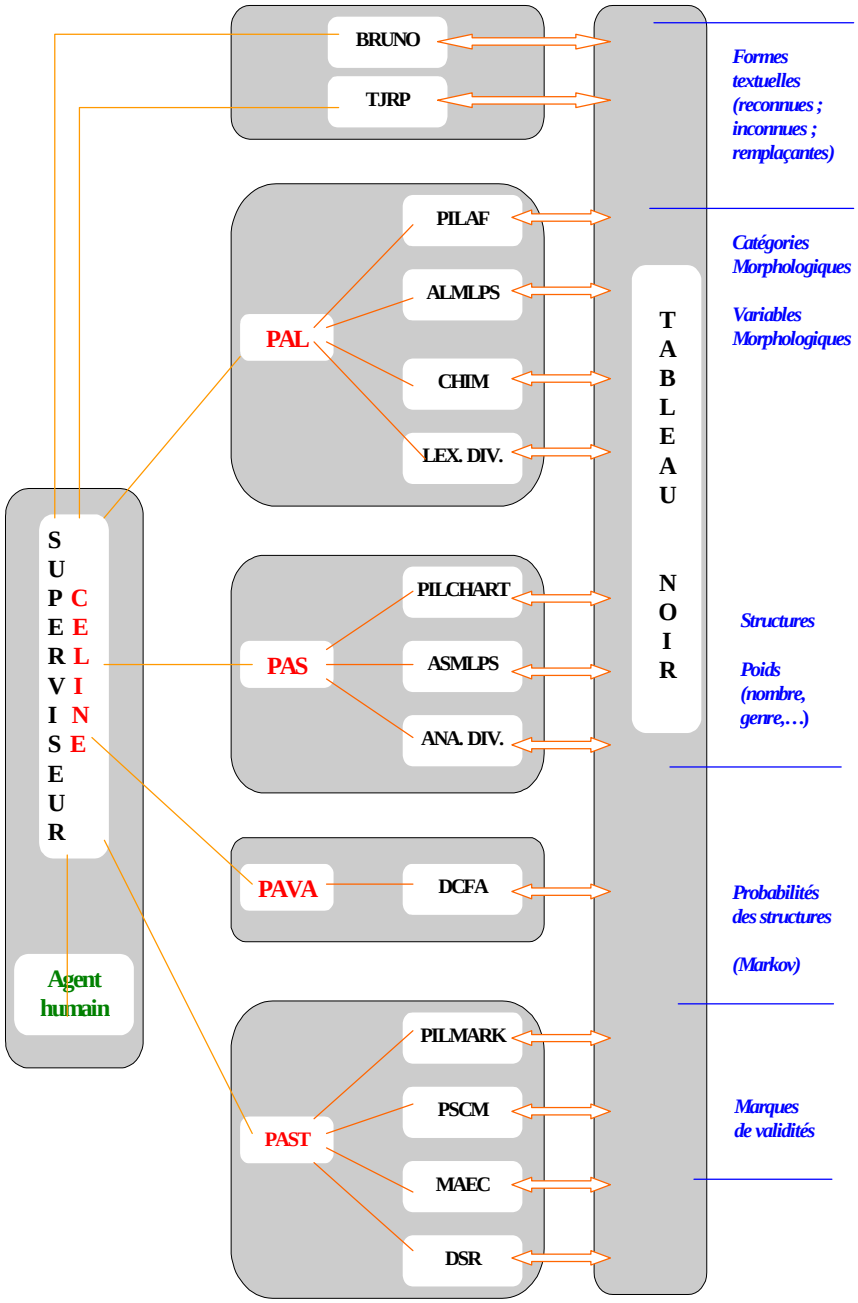


Figure 9.6 : le système CELINE à un seul niveau

9.6.3 CLASSIFICATION SUPPLEMENTAIRE DES AGENTS LEXICAUX EXTÉRIEURS

En imaginant un ensemble de plusieurs dizaines, centaines de lexiques couvrant les différents domaines, nous avons proposé une méthode de recherche par lexico-thésaurus que nous ne reprendrons pas ici.

Un problème de classification de ces agents lexicaux existe car les accointances des agents de contrôle concernés doivent inclure des renseignements du type : l'agent peut - il faire de la génération, l'agent peut-il proposer une forme textuelle remplaçante pour

une forme textuelle inconnue, etc. La classification proposée permet de codifier ce type de renseignements.

De même que dans le domaine des anti-virus, il existe des logiciels meilleurs pour la détection des virus et d'autres meilleurs pour leur traitement, il est logique que pour des lexiques concurrents, certains soient plus efficaces en reconnaissance et d'autres plus efficaces en proposition. La mise en concurrence des agents devrait permettre une prise en compte de cet aspect.

Cette classification pourrait voir une extension à l'ensemble des agents du système, extension inutile pour l'instant.

9.6.3.1 Attributs du classement des lexiques

La désignation des agents du secteur lexical débute par un L.
La deuxième lettre correspond à la propriété décrite, les lettres suivantes aux différents cas possibles.

> Exemple :
> LRM : agent du secteur lexical permettant la Reconnaissance de Mot isolé uniquement

Langue :
Nous n'envisagerons que la langue française. Les mots étrangers occasionnels seront traités comme appartenant à un domaine particulier.

Type de reconnaissance

1. LRM de mot isolé ou
2. LRG de groupe de mots

Fourniture de traits

1. LT0 Simple reconnaissance de graphie (sans délivrance d'aucun renseignement morphologique, sémantique)
2. LTM traits morphologiques (genre, nombre, ...)
3. LTS traits sémantiques
4. LTP traits pragmatiques
5. multi-traits LTMS (morphologique et sémantique),
6. LTSP (sémantique et pragmatique) etc

Possibilités de générations

1. LG0 : sans génération
2. LGA : automatique (le module recherche lui-même les renseignements dont il a besoin)
3. LGG : guidé (les renseignements utiles sont fournis à l'agent générateur)

Possibilité de corrections intégrées

1. LC00 pas de correction
2. Correction non contextuelle
   LC0E par essais multiples (traitement orthographique),
   Par similarité
   - LC0SS clé squelettes,
   - LC0SP clés phonétiques,
   - LC0SD calcul de distances,

Par interprétation de variables
- LC0IM morphologique,
- LC0IS sémantique,
- LC0IM pragmatique,

3. Correction contextuelle
   - LCCG sur la graphie,
   - LCCM sur les variables morphologiques,
   - LCCS sur les variables sémantiques

9.6.3.2 Tables et structure de données relatives aux lexiques

Identifiant	Nom du lexique	Domaine	Type de reconnaissance	Fourniture de traits	Génération	Correction intégrée
000	PILAF	Général	LRP	LTM	LGGM	LC00

9.6.3.3 Les agents lexicaux utilisés

N° agent	*Nom*	*Domaine*	*Particularités*
5	ALMLPS	Dictionnaire(s) personnel(s)	Agent dont les connaissances sont déterminées par apprentissage.
3	CHIMOR	Chimie Organique	Automate d'états finis de reconnaissance de la nomenclature (début de développement). La catégorie morphologique associée est unique : substantif
4	ELEC	Électricité	Recueil de termes techniques. Les mots ont une catégorie et des variables morphologiques associées. Exemples : diode (subc, fem, sin), diodes (subc, fem, plu). amorti(adjq, mas, sin)
2	NONPRO	Noms propres	Au moins quelques caractères en majuscule.
1	PILAF	Vocabulaire courant	Agent lexical présenté en annexe

9.7 CONCLUSION

Notre système repose sur une hypothèse plausible : une multiplicité d'agents spécialistes développés par des équipes différentes.
La non disponibilité de ces agents a été une gêne considérable pour la réalisation, malgré une compensation par un certain nombre de simulations.

Trois maquettes ont été réalisées :

1) Une première maquette sur Macintosh comprenant quatre agents :
   - l'agent lexical PILAF,
   - l'agent d'analyse syntaxique PILCHART,
   - un agent d'analyse syntaxique déterminant les structures,
   - l'agent DCFA de correction et vérification des accords.

Ces agents correspondent par message déposé dans une boite à lettres mais le parallélisme était simulé et l'ordonnancement séquentiel.

Dans l'annexe D nous donnons une trace d'exécution montrant la construction progressive de la solution vue à travers l'évolution des structures. Le tableau noir simplifié étant simulé par cet ensemble de structures.

2) La maquette d'étude du lexico-thésaurus.

3) CELINE sur machine UNIX ou PC comprenant les agents décrits dans ce chapitre

Nous comptons poursuivre le développement de CELINE en introduisant un objectif pédagogique d'aide au rédacteur ou d'apprentissage de la langue et en maintenant notre conception d'un système lourd générateur d'applications légères adaptées à des utilisateurs. Le chapitre de conclusion va permettre de revenir sur ces points.

10. CONCLUSION

La *correction des erreurs* fait intervenir la langue, le mode d'expression (écrit ou oral), le rédacteur et le système matériel assurant l'interface entre le texte mental et le texte électronique. Ce quadruple niveau multiplie le nombre d'approches possibles ainsi que le nombre et la complexité des difficultés à résoudre.

Différentes modalités de correction peuvent s'envisager, en commençant par les possibilités d'une correction automatique ou d'une correction interactive avec le rédacteur. Ce choix introduit l'aspect cognitif de la prise de décision face à un inattendu représentant une suspicion de fautes.

Par rapport à ce contexte très général, le sujet de thèse proposé éliminait un certain nombre de difficultés en limitant le contexte à celui d'un rédacteur utilisant son traitement de texte, avec pour conséquence :

- La dispense d'une prise en charge complète du domaine de l'oral[12]
- la limitation des erreurs liées aux interfaces Homme-Machine en ne considérant que les interfaces du type clavier.

La finalité de notre travail, dans le cadre de la correction des erreurs, était la recherche d'une approche permettant de prendre en compte aussi bien les aspects contextuels que l'aspect multi-niveaux dans les prises de décision du système.

Notre direction de recherche devait diverger des voies prises pour les correcteurs commerciaux qui possédaient sur leur terrain, une avance certaine et irrattrapable. Dans le chapitre d'introduction, nous avons déjà vanté l'efficacité de WORD 97 et les progrès par rapport aux versions antérieures.

Nous avons alors ajouté dans le cadre défini quatre objectifs à long terme :

1. se rapprocher d'un système robuste offrant la meilleure couverture linguistique possible,
2. étudier la possibilité d'un système complètement automatique en estimant l'objectif précédant au moins partiellement atteint,
3. accepter, en attendant mieux, des agents réels imparfaits,
4. concevoir un système « boite à outils » pouvant s'orienter vers des applications pédagogiques telle que l'enseignement de l'orthographe ou des applications pratiques du domaine de l'industrie de la langue telles que l'apprentissage de la langue, l'aide au rédacteur, etc.

Une autre dominante constituant un cinquième objectif à court terme est venue se greffer. Dans l'immense majorité des systèmes humains amenés à prendre des décisions, ces décisions font intervenir un aspect quantitatif, seul capable de pouvoir comparer objectivement des solutions concurrentes :

5. Prises de décision appuyées par un aspect quantitatif.

La problématique des langues naturelles avec, en particulier, le morcellement des connaissances et la problématique de la correction d'erreurs a guidé notre démarche dans la recherche d'une solution.

---

[12] Mais un texte peut inclure des dialogues et de ce fait l'oral peut s'introduit sous la forme d'un domaine secondaire.

La recherche d'un système universel de grande robustesse quel que soit le rédacteur nous a entraîné sur la piste du multi-modules, avec l'hypothèse simplificatrice pour un rédacteur particulier du *modèle linguistique partiel suffisant*.

Le multi-modules pouvait se résoudre par une architecture de type client-serveur mais la nécessité de décisions contextuelles multi-niveaux nous a lancé sur la voie *d'une architecture multi-agents*. Concernant le mode de communication, notre option fondamentale a été de choisir une communication par *tableau noir parallèle* pour la tâche principale de correction des erreurs. Une communication directe par message complète le premier mode pour des tâches comme l'activation d'agents et certains échanges d'informations. Certains points de l'architecture (pilotes, communications secondaires par messages) ont été justifiés par le souci d'une optimisation dans la gestion du flot d'informations au niveau du tableau noir.

La visée d'un système de correction automatique change radicalement les stratégies possibles par rapport à un système interactif. Aucune possibilité de solution ne peut alors être spontanément rejetée et nous avons défini un ensemble de probabilités ou de coefficients permettant d'affecter un poids à chacune des solutions en satisfaisant ainsi nos objectifs 2, 3 et 5. La recherche du maximum de solutions augmente considérablement le coût au niveau combinatoire mais nous avons admis en postulat que les performances croissantes des machines pourraient compenser cet alourdissement. Nous avons largement utilisé les modèles de Markov et, dans le cas particulier de la vérification des accords, nous avons ébauché la théorie des structures.

La recherche d'une correction automatique n'est pas contradictoire avec celle d'une correction interactive faisant appel aux connaissances de l'utilisateur. Les efforts en vue d'une correction automatique et d'une quantification ont été payants dans le cadre d'une correction interactive par une meilleure sélectivité des solutions concurrentes proposées à un rédacteur. Sur le plan de l'architecture du système, l'utilisateur peut conceptuellement être envisagé sous la forme d'un agent du système dénommé agent humain. Le degré d'interactivité peut alors être déterminé par un choix de coefficients appropriés (par exemple un coefficient de crédibilité de l'agent humain initialisé à zéro fait passer de fait le système en mode automatique).

Les objectifs fixés ont conduit à la nécessité de bien connaître le rédacteur et le système par auto-apprentissage. La connaissance du rédacteur permet d'entrevoir des applications, principalement pédagogiques, dans le cadre des industries de la langue correspondant à notre quatrième objectif.

L'équipe TRILAN étant sans expérience sur le plan des systèmes répartis et du multi-agents, il a fallu intégrer une partie de cette connaissance et de ce savoir-faire et acquérir en quelque sorte une mentalité d'approches différentes.

Le système final possède deux facettes distinctes et une option fondamentale :

- Une « grosse machine » éventuellement répartie, renfermant l'ensemble du savoir linguistique, permettant une grande robustesse, et faisant office de génératrice de systèmes individualisés.
- Un système léger implanté sur le site de l'utilisateur, système devenant, au fur et à mesure de l'apprentissage, de plus en plus autonome et donc indépendant de la machine centrale.

- Intégration à la demande de l'agent humain permettant une correction interactive soit sur le site de l'utilisateur (le rédacteur) soit sur la machine centrale (correcteurs professionnels d'une société de service).

Ce double aspect coïncide avec une solution futuriste envisagée par de nombreux développeurs professionnels misant sur les réseaux. D'une manière générale, les logiciels deviennent de plus en plus complexes et demandent de la part des fabricants de gros investissements se traduisant par des prix de vente élevés incompatibles avec une utilisation non professionnelle. Ce prix élevé explique en partie le piratage. L'idée est donc de faire payer l'utilisation du logiciel et non le logiciel (un peu comme les chaînes payantes de film pour la télévision). Pour cela :

- Un ordinateur du réseau fait office de serveur de logiciels.
- Les logiciels ne sont plus vendus mais loués à chaque utilisation.
- Chaque utilisateur ne paye que ce qu'il consomme (en téléchargeant à chaque utilisation le logiciel ou les parties du logiciel nécessaires et en payant en fonction par exemple de la durée d'utilisation).
- Bien entendu cette solution ne devient possible qu'avec une amélioration des vitesses de transmissions des réseaux et en réduisant les besoins de transferts.

Le système CELINE se moule bien dans cette approche : connexion obligatoire à l'ordinateur central (les royalties …). Les transferts sont minimisés : seuls l'activation du logiciel implanté sur le site utilisateur et les recherches de renseignements ne faisant pas partie du MPLS de l'utilisateur implique une utilisation du réseau.

Pour terminer les aspects positifs, je soulignerai le fait que les divers chapitres spécifiques de CELINE ont fait l'objet de communications à des colloques avec comité de lecture. Ces colloques ont permis d'ouvrir mon horizon de recherche.

Sur le plan des regrets, nous pouvons citer :

1. Notre volonté d'une couverture maximale n'a pas permis de faire intervenir le niveau sémantique qui commence à n'être défini que dans quelques rares domaines et à n'être modélisé que de manière trop hétérogène pour pouvoir être intégrable. La seule possibilité actuelle passerait par l'intermédiaire d'agents traitant complètement une partie d'un corpus relative au domaine de l'agent.
2. Le travail de développement étant l'œuvre d'une seule personne (en dehors de PILAF), a permis de prototyper plus ou moins complètement une vingtaine d'agents. De ce fait, certaines expérimentations en vraie grandeur n'ont pu être encore menées.
3. Les aspects cognitifs d'un certain nombre d'agents demandent d'être approfondis.
4. La durée de réalisation de cette thèse, durée imposée par mes obligations professionnelles à temps plein a rendu moins innovante la conception globale du système.

Pour résumer : nous avons commencé à mettre en place un système semi-automatique permettant la correction assistée par ordinateur. Ce système ne dispose pas, pour l'instant, d'agents extérieurs suffisamment compétents. L'évolution actuelle de la communication avec le réseau INTERNET, les progrès attendus sur le plan des lexiques

et des analyseurs divers traités en tant que modules isolés feront qu'au fil du temps un système comme CELINE pourra être pleinement opérationnel et qu'ainsi les problèmes que nous avons commencés à étudier se poseront. Ces futurs agents évolués disposeront alors d'un ensemble d'accueil au niveau de leurs performances.

BIBLIOGRAPHIE

(Aït Kaci 89a) Hassan Aït Kaci, Roger Nasr. *Integrating logic and functional programming.* Lisp and Symbolic Computation 2 pp. 51-89 - 1989.

(Aït Kaci 89b) Hassan Aït Kaci. *Efficient implementation of lattice opération.* ACM Transactions on programming languages and systems 11 :1 pp 116-146 - 1989.

(Allen 87) James Allen. *Natural Language Understanding.* The Benjamin/Cummings Publishing Compagny, Inc - 1987.

(Amselem 91) Denis Amselem. *Compréhension d'un texte en langue naturelle : Analyse et Inférence de Plans.* Rapport de DEA d'informatique - Université Joseph Fourier - U.F.R. Informatique & Mathématiques Appliquées -1991.

(Antoniadis 84) Georges Antoniadis, *Elaboration d'un système morpho-syntaxique d'une langue naturelle. Application en informatique documentaire.* Thèse de troisième cycle, Université des sciences sociales Grenoble - 1984.

(Antworth 90) Evan L. Antworth. *A Two-level Processor for Morphological Analysis.* Academie Computing Departement - Summer Institute of Linguistic -7500 W Camp Widson Road Dallas, TX 75236 USA - 1990.

(Antoine 94) Jean-Yves Antoine. *Coopération syntaxe-sémantique pour la compréhension automatique de la parole spontanée.* Thèse de doctorat, Institut National Polytechnique de Grenoble. - Décembre 94.

(ATALA 94) Revue semestrielle de l'ATALA *Traitement automatique des langues naturelles : Morphologie computationnelle.* Volume 35 n°2, édition TAL, Jussieu. 1994.

(ATALA 95) Revue semestrielle de l'ATALA *Approche probabiliste de l'analyse syntaxique. Traitements probabilistes et corpus* -Volume 36 - 1995 -

(ATALA 97a) Revue semestrielle de l'ATALA *Traitement automatique des langues naturelles : Prosodie et syntaxe.* Volume 38 n°1, édition TAL, Jussieu. 1997.

(ATALA 97b) Revue semestrielle de l'ATALA *Traitement automatique des langues naturelles : Etat de l'art.* Volume 38 n°2, édition TAL, Jussieu. 1997.

(Baujard 94) O. Baujard, C. Spinu, C. Garbay, S. Pesty and J.M. Chassery, *COALA: Un Langage pour la Conception de Systèmes Adaptatifs de Résolution de Problèmes*. Colloque langage a objets, Grenoble 1994.

(Bellet 96) F. Bellet, C. Garbay. *Des processus adaptatifs et coopératifs pour la vision bas niveau* Actes du 10ième Congrès "Reconnaissance des Formes et Intelligence Artificielle", Vol. 2, pp. 516-526, AFCET / AFIA, 1996.

(Bellissant 87) Camille Bellissant. *Méthodes syntaxiques en reconnaissance des formes.* Cours polycopié DEA d'Informatique Option Intelligence Artificielle - Laboratoire TIM3 - IMAG - 1987.

(Bernadet 94) Maurice Bernadet. *Utilisation d'un système distribué de maintien de cohérence entre croyances pour la coopération entre agents.* Actes des Deuxièmes Journées Francophones Intelligence Artificielle Distribuée & systèmes multi-agents - Voiron - Mai 94

(Bescherelle 84) Bescherelle. *Le Nouveau Bescherelle - 3 La grammaire pour tous.* Librairie Hatier, Paris - Mai 1984.

(Boissier 93) O. Boissier. *Problème du contrôle dans un système intégré de vision* Thèse de doctorat - Laboratoire du LIFIA, IMAG - Janvier 93.

(Bouabdallah 91) A Bouabdallah. *Problèmes liés aux systèmes parallèles et distribués.* Thèse de doctorat. Université de Paris-Sud. - Décembre 91.

(Bouchaffra 93) Djamel Bouchaffra, Geneviève Lallich-Boidin et Jacques Rouault. *Echantillonnage stratifié avec bootstrap : application à l'analyse morphologique.* Secondes journées internationales d'analyse statistique de données textuelles. Montpellier - Octobre 93.

(Bouchard 91) L Bouchard, L. Emirkanian, D. Estival, C. Fay-Varbier, C. Fouquere, G. Prigent. *Environnement de Génie Linguistique.* Rapport de recherche - Projet GDE - CNET Lannion - Octobre 91.

(Boucher 96) A. Boucher, C. Garbay. *Segmentation de séquences d'images cytologiques par un système multi-agents,* Actes des 4ièmes Journées Francophones IAD & Systèmes Multi-Agents, (J.P. Müller & J. Quinqueton, eds), pp. 125-135,Hermès, 1996.

(Boucher 98) A Boucher, C. Garbay, *Des agents spécialisés pour la compréhension de séquences d'images.* Actes du 11ièm Congrès "Reconnaissance des Formes et Intelligence Artificielle" (RFIA'98). Vol. II, pp. 275-284.AFCET / AFIA, 1998.

(Boury-Brissey 94) Anne-Claire Boury-Brisset, Bernard Moulin. *Mise en œuvre de raisonnements multiples dans un système multi-agents.* Actes des Deuxièmes Journées Francophones Intelligence Artificielle Distribuée & systèmes multi-agents - Voiron - Mai 94.

(Bouzouane 94) A. Bouzouane, D. Boulanger, B.T.David, C.Vial. *Un modèle multi-agents basé sur le tableau noir : Modèle Multi-mondes.* Actes des Deuxièmes Journées Francophones Intelligence Artificielle Distribuée & systèmes multi-agents - Voiron - Mai 94

(Caillaud 97) Bernard Caillaud. *Apprentissage de connaissances prosodiques pour la reconnaissance automatique de la parole.* Thèse de doctorat. Institut National Polytechnique de Grenoble. – Décembre 1996.

(Calliope 89) Ouvrage collectif. *La Parole et son traitement automatique.* Masson - Collection Technique et Scientifique des Télécommunications –1989.

(Carbonnel 81) J. G. Carbonnel, P.J. Hayes. *Dynamic Strategy Selection in Flexible Parsing.* 19th Annual meeting of the ACL – 1981.

(Carbonnel 83) J. G. Carbonnel, P.J. Hayes. *Recovery Strategies for Parsing Extragrammatical Language.* AJCL 9:3-4, - 1983 pp 123 - 146.

(Carbonnel 84) J. G. Carbonnel, P.J. Hayes.. *Coping with Extragrammaticality.* 19th Coling, Stanford, USA, July 1984, pp 437 - 443.

(Carlier 88) J. Carlier , P. Chrétienne. *Problèmes d'ordonnancement : modélisation, complexité, algorithmes.* Masson, Paris. - 1988.

(Caromel 91) D. Caromel. *Programmation Parallèle Assynchrone et impérative : étude et propositions.* Thèse de doctorat. Université de Nancy I. - Février 1991.

(Catach 80 a) N. Catach. *L'orthographe française.* Edition Nathan - 1980.

(Catach 80 b) N. Catach, D. Duprez, M. Legris. *Enseignement de l'orthographe, Alphabet phonétique, Typologie des fautes.* Edition Nathan - 1980.

(Catach 81) N. Catach. O*rthographe et lexicographie* Edition Nathan - 1980.

(Cerisara 96) Christophe Cerisara. *Un modèle hybride pour la reconnaissance automatique de la parole*. Rapport de DEA. IMAG. - Juin 1996.

(Chanod 98) J.P. Chanod. *TALN : recherche et industrie* – Revue semestrielle de l'ATALA – Volume 38, n° 2, pp. 135-144 – Janvier 1998.

(Chomsky 65) Noam Chomsky. *Aspects of the theorie of syntax.* MIT Press. -1965.

(Chomsky 87) Noam Chomsky. Traduction : *La nouvelle syntaxe.* Editions du seuil, Paris - 1987.

(Cohard 88) Brigitte Cohard. *Logiciel de détection et de correction des erreurs lexicales.* Mémoire d'ingénieur CNAM- Centre régional associé de Grenoble - Mars 1988.

(Comer 89) Douglas E. Comer, David L. Stevens. *Internetworking with TCP/IP-Volume III - Client-Serveur Programming and Applications.* Prentice Hall – 1989.

(Corkill 89) D.D. Corkill. *Advanced architectures : concurency and Parallelism.* Blackboard architectures and applications. Academic Press. - 1989.

(Courtin 77) Jacques Courtin. *Algorithme pour le traitement interactif des langues naturelles.* Thèse de doctorat d'état. USMG - INPG - Grenoble - Octobre 1977.

(Courtin 90) J. Courtin, D. Dujardin, D. Genthial, I. Kowarski, V.L. de Lima. *Vers un système complet de détection correction.* Rapport de Recherche - IMAG - Mai 1990.

(Courtin 91) J. Courtin, D. Dujardin, D. Genthial, I. Kowarski, V.L. de Lima. *Towards a complete detection/correction system.* International conference on current issues in computational linguistics. Penang, Malaysia, pp 158-173. June 1991.

(Courtin 92) J. Courtin, D. Dujardin, D. Genthial, I. Kowarski. *Outils lexicaux de l'équipe TRILAN ; bilans et perspectives.* Les actes des journées GRECO-PRC. Communication Homme Machine - Séminaire LEXIQUE - Toulouse - Janvier 1992.

(Courtin 95) J. Courtin, D. Dujardin, D. Genthial, I. Kowarski. *Analyse et génération morphologique avec le système PILAF*. TAL Revue semestrielle de l'ATALA Volume 35 n°2. pp 92-108 - 1995.

(Crochemore 93) M. Crochemore, D. Perrin, J.-E. Pin. *Les automates finis.* La recherche en informatique, Le courrier du CNRS N°80. - Février 1993 .

(Dabéne 89) Louise Dabéne. *Stratégies d'apprentissage et didacticiels de langues. Contribution à une évaluation.* Centre de didactique des langues. Université Grenoble 3. - 1989 .

(Damereau 64) F.J. Damereau. *A technique for computer detection et correction of spelling errors.* Communication of the ACM, Vol 7, pp 171-176, Mars 1964.

(Debili 77) Fathi Debili. *Traitements syntaxiques utilisant des matrices de précédence fréquentiellement construites automatiquement par apprentissage.* Thèse de docteur-ingénieur. Paris VII – 1986.

(Debili 82) Fathi Debili. *Analyse syntaxico-sémantique fondée sur une acquisition automatique de relations lexicales-sémantiques.* Thèse d'état. Paris XI – 1982.

(Debili 86) Fathi Debili. *Le traitement des entrées non prévues dans les systèmes de compréhension du langage naturel : détection et correction des graphies fautives* Rapport d'activité du PRC Communication Homme-Machine, Pôle langage naturel.Orsay – 1986.

(De Loupy 90) Claude De Loupy. *La méthode d'étiquetage d'Eric BRILL* Traitements probabilistes et corpus -Volume 36 - Revue semestrielle de l'ATALA – 1995.

(Demazeau 92) Yves Demazeau. *Modèle d'Agent Cognitif Hiérarchique (COHIA).* Rapport interne du LIFIA - 1992.

(Demazeau 95) Yves Demazeau. *From cognitive interactions to collective behaviour in agent-based systems.* European Conference on Cognitive Science. Saint Malo – Avril 1995.

(Dinnematin 90) Seymour Dinnematin, Didier Santz. *Sept correcteurs pour l'orthographe et la grammaire.* SVM - Décembre 92.

(Earley 70) J.C. Earley. *An efficient context-free parsing algorithm.* Proceedings of the ACM. Février 1970.

(Erman 80) L.D. Erman, F. Hayes-Roth and V. Lesser. *The HERSAY II speech understanding System : integration knowledge to resolve Uncertainty.* ACM Computing Surveys. June 1980.

(El-Beze 95) Marc El-Beze. *Intégration de Contraintes Syntaxiques dans un Système d'Etiquetage Probabiliste.* Traitements probabilistes et corpus -Volume 36 - Revue semestrielle de l'ATALA - 1995.

(Fennell 77) R.D. Fennell, V.R. Lesser. *Parallelism in Solving : a case study of Hearsay-II.* IEEE Transaction on Computers. Février 1977.

(Ferber 89) Jacques Ferber. *Objets et agents : Une étude des structures de représentation et de communication en intelligence artificielle.* Thèse de doctorat d'état - Université de Paris VI - 1989.

(Ferber 95) Jacques Ferber. *Les systèmes Multi-agents. Vers une intelligence collective.* InterEditions, Paris – 1995.

(Fouquere 88) Christophe Fouquere. *Systèmes d'analyse tolérants de langage naturel.* Thèse de doctorat - Université de Paris Nord - Janvier 1988

(Frechet 92) A.L. Frechet, D. Dujardin, J. Caelen, J. Courtin, M.A. Morel. *Analyse Lexicale pour le français oral avec le logiciel PILAF*. Les actes des journées GRECO-PRC. Communication Homme Machine - Séminaire LEXIQUE - Toulouse - Janvier 92.

(Garbay 98) C. Garbay, F. Bellet, A. Boucher. *Des agents situés pour l'interprétation de scènes*, Revue d'Intelligence Artificielle, Numéro spécial "Intelligence Artificielle Distribuée / Systèmes Multi-Agents" - 1998

(Gardent 95) C Gardent, K. Baschung *Techniques d'analyse et de génération pour la langue naturelle* Edition Adosa – 1995.

(Gazdar 89) G. Gazdar, C. Mellish. *Natural Language processing in LISP. An introduction to computational linguistics* Addison-Wesley Publishing Compagny. - 1989.

(Genthial 91) Damien Genthial. *Contribution à la construction d'un système robuste d'analyse du français.* Thèse de doctorat - Université Joseph Fourier - Grenoble - Janvier 1991-

(Genthial 96) D. Genthial, J. Courtin, J. Menézo. *Distributing and porting general linguistic tools . Internationnal Conference on Computational Linguistics* – COLING 96 – University of Copenhagen, Danemark. Août 1996-

(Gilloux 91) Michel Gilloux. *Apprentissage Automatique de transducteurs de Mots.* Les actes des journées GRECO-PRC. Communication Homme Machine - Toulouse - Janvier 91.

(Grandchamp 86) J.-M. Grandchamp, *Processus de rattrapage d'erreurs sur les mots inconnus*, rapport de stage de DEA. Université Paris XI - 1986.

(Granger 83) R.H. Granger. *The NOMAD system : Expectation Based Detection and Correction of Errors during Understanding of Syntactically and Semantically Ill-Formed Test.* AJCL 9:3-4, 1983, pp 188 - 196.

(Gross M 75) M. Gross *Méthodes en syntaxe* Edition Herman 1975

(Gross M 75) M. Gross, A. Lentin. *Notion sur les grammaires formelles.* Edition Gauthier-Villars Paris. -1967.

(Habert 95) Benoît Habert, André Salem. *L'utilisation de catégorisations multiples pour l'analyse quantitative de données textuelles.* Traitements probabilistes et corpus -Volume 36 - Revue semestrielle de l'ATALA – 1995.

(Haton 91) J.P. Haton, J.M. Pierrel, G. Perennou, J. Caelen, J.L. Gauvain. *Reconnaissance automatique de la parole.* DUNOD Informatique - 1991.

(Haton 91) J.P. Haton, N.Bouzid, F.Charpillet, M.C. Haton, B. Lâasri. *Le raisonnement en intelligence artificielle. Modèles, techniques et architectures pour les systèmes à bases de connaissances.* IIA - InterEditions - 1991.

(Hirschman 91) L. Hirschman, S. Senef, D. Goodine, M. Philips. *Integrating Syntax and Semantics into Spoken Language Understanding*. Massachusetts Institute of Technology. DARPA meeting - Mars 1991.

(Jayez 79) Jacques-Henri Jayez. *Une approche de la compréhension par machine du langage naturel.* Thèse d'état, Paris VII - 1979.

(Jayez 82) Jacques-Henri Jayez. *Compréhension automatique du langage naturel, le cas du groupe nominal en français.* Masson- collection Méthode + programmes - 1982.

(Jensen 83) K. Jensen, G.E. Heudorn, L.A. Miller, Y. Ravin. *Parse Fitting and Prose Fixing : Gettind a Hold on Ill-formedness.* AJCL 9:3-4, 1983, pp 147 - 160

(Kallas 87) Ghassan Kallas. *Résolution des solutions multiples en Analyse Morphologique Automatique de Langues Naturelles. Utilisation des Modèles de Markov*. Thèse de doctorat d'état. Centre de Recherche en Informatique Appliquée aux Sciences Sociales - Grenoble - Juin 1987.

(Kaplan 73) R.M. Kaplan. *A general syntactic processor in Natural langage processing (R. Rustin)* Algorithmics Press, New York. – 1973.

(Kay 81) Martin Kay. *Unification grammars.* Xerox publication. - 1981.

(Kay 84) Martin Kay. *Functional unification grammar : a formalism for machine translation* 10 th Coling, Standford, USA, pp 75-78. - July 1984.

(Kriouile 90) A. Kriouile. *La reconnaissance automatique de la parole et les modèles markoviens cachés. Modèles du second ordre et distance de Viterbi à optimalité locale.* Thèse de l'université de Nancy I. 1990.

(Lâasri 89) H. Lâasri, B Maître. *Organisation du contrôle dans les architectures de blackboards : étude, évaluation, comparaison.* Revue d'intelligence artificielle, Volume 3 pp.105-146 - 1989.

(Lallich-Boidin 86) Geneviève Boidin-Lallich. *Analyse syntaxique automatique du français : application à l'indexation automatique.* Thèse de doctorat. . Université des Sciences sociales de Grenoble.- Octobre 1986.

(Lallich-Boidin 98) Geneviève Boidin-Lallich. *Communication homme-machine et recherche d'information fondées sur le traitement automatique de la langue : application, bilan, perspectives.* Mémoire en vue de l'habilitation à diriger des recherches. Université Stendhal – Grenoble 3. - Février 1998.

(Laporte 98) E. Laporte. *Les mots : un demi-siècle de traitements.* Revue semestrielle de l'ATALA – Volume 38, n° 2, pp. 47-68 – Janvier 1998.

(Lesmo 84) Leonardo Lesmo, Pietro Torasso. *Interpreting Syntactically Ill-Formed Sentences.* 10th Coling, Stanford, USA, July 1984, pp 534 - 539.

(Letellier 93) Sabine Letellier. *ECLAIR, un sytème d'analyse et de correction lexicales multi-experts et multi-lexiques.* Thèse de doctorat - Université de Paris-sud - Centre d'Orsay - Décembre 1993.

(Levenshtein 66) V.I. Levenstein *Binary codes capable of correcting deletions, insertions or reversals*. Soviet Physics Doklady. pp 707-710 - 1966

(Lowrance 75) R. Lowrance, R. Wagner *An extension of the string-to string correction problem* Journal of the A.C.M. Vol 22 n° 2 pp 177-183 – 1975.

(Lu 86) Lu Chengren. *Détection et correction des erreurs dans un texte écrit en langue naturelle.* Rapport de DEA Grenoble I. - 1986.

(Luzatti 95) D. Luzatti *Le dialogue verbal Homme-Machine* Edition Masson 1995

(Mangeot Lerebours 97) Mathieu Mangeot Lerebours. *Outils pour lexicographes naïfs*. DEA d'informatique systèmes et communication Edition DUNOD. IMAG. - 1967.

(Maurel 89) Denis Maurel. *Reconnaissance de séquences de mots par automates, adverbes de date du français.* Thèse de doctorat en informatique - Université de Paris VII – 1989.

(Marcus 67) S. Marcus. *Introduction mathématique à la linguistique structurale* – Edition DUNOD – 1967.

(Margue 95) Anne Margue. *Intégration d'outils linguistiques portables et réutilisables.* Mémoire d'ingénieur C.N.A.M. - Grenoble - Décembre 1995.

(Mel'cuk 93) I. Mel'cuk, *Cours de morphologie générale Vol 1, 2 & 3.*Les presses de l'université de Montréal, CNRS édition CNRS – Novembre 1993.

(MEN 85) Ministère de l'éducation nationale, Mission aux technologies nouvelles, *Informatique pour tous*. . – CNDP et MEN. Août 1985.

(MEN 97a) Ministère de l'éducation nationale, *Programmes de l'école primaire : des apprentissages premiers aux apprentissages fondamentaux et aux approfondissements.* CNDP et MEN. - 1997.

(MEN 97b) Ministère de l'éducation nationale, *Evaluation à l'entrée en classe de sixième.* CNDP et MEN. - 1997.

(MEN 97c) Ministère de l'éducation nationale, *Evaluation à l'entrée en classe de seconde.* CNDP et MEN. - 1997.

(Menézo 92 a) Jacques Menézo, Jacques Courtin, Damien Genthial. *Désambiguïsation lexicale par filtrages en cascade.* Les actes des journées GRECO-PRC. Communication Homme Machine - Séminaire LEXIQUE - Toulouse - Janvier 1992.

(Menézo 92 b) Jacques Menézo. *Désambiguïsation lexicale par filtrages en cascade.* Mémoire d'ingénieur C.N.A.M. - GRENOBLE - Juin 1992.

(Menézo 94) Delphine Menézo. *Etude et réalisation d'un analyseur syntaxique des langues naturelles utilisant la méthode des cartes.* Mémoire d'ingénieur C.N.A.M. - GRENOBLE - Mars 1994.

(Menézo 96 a) Jacques Menézo, Damien Genthial, Jacques Courtin. *La méthode des structures, principe et mise en œuvre dans CELINE,* TALN'96, Marseille, Mai 1996.

(Menézo 96 b) Jacques Menézo. *Reconnaissances pluri-lexicales dans CELINE un système multi-agents de détection-correction des erreurs*, NLP+IA 96, Moncton, Canada, Juin 1996.

(Menézo 96 c) Jacques Menézo. *CELINE un système multi-agents de détection-correction des erreurs lexicales et syntaxiques*, CLIM'96, Montréal, Canada, Juin 1996.

(Menézo 96 d) Jacques Menézo. *Le modèle de l'utilisateur comme modèle linguistique partiel suffisant en détection-correction des erreurs"* - Colloque ILN.'96 - Nantes - Octobre 1996 -

(Menézo 97) Jacques Menézo. *Problématique et réalisation d'un système multi-agents de TALN à deux niveaux de modélisation (complète et partielle suffisante).* Acte de FRACTAL-97- pp. 303-312 - Besançon - Décembre 1997.

(Menézo 98) Jacques Menézo, Damien Genthial, Jacques Courtin. *Modèles humains dans un système multi-agents orienté apprentissage et détection correction des erreurs.* NLP+IA 98, Moncton, Canada, Août 1998.

(Minsky 91) Naftaly Minsky. *Law-governed systems.* Software Engineering Journal, Volume 6 - Septembre 1991.

(Nii 86) H.P. Nii. *The blackboard Model of Problem Solving.* The AI Magazine. - Août 1986.

(Occello 91) M. Occello, J.Y. Tigli. *Graphique interactif distribué pour le contrôle en robotique mobile.* Actes de la conférence régionale sur l'informatique des hautes technologies. - Marseille - Novembre 1991.

(Occello 93) Michel Occello. *Blackboards Distribués et Parallèles : Application au Contrôle de Systèmes Dynamiques en Robotique et en Informatique musicale.* Thèse de Doctorat. - Université de Nice-Sophia Antipolis - Janvier 1993.

(Oquendo 94) Flavio Oquendo, Ilham Alloui, Selma Arbaoui, Bernard Debord, Safia Larous, Guy Tassart. *SCALE/ALPS : un générateur de systèmes multi-agents pour la construction d'environnements centrés processus assistés par ordinateur.* Actes des Deuxièmes Journées Francophones Intelligence Artificielle Distribuée & systèmes multi-agents - Voiron - Mai 94

(Pécate 92) Jean-Marie Pécate. *Tolérance aux fautes dans les interfaces homme machine. Traitement des chaînes phonétiques, des chaînes orthographiques et des structures syntaxiques* . Thèse de doctorat d'état. Université Paul Sabatier Toulouse - Janvier 1992.

(Pérennou 86) Guy Pérennou, P. Daubeze, F. Lahens. *La vérification et la correction automatique de textes : le système VORTEX.* Techniques et sciences informatiques, pp 285-304. - 1986.

(Pérennou 90) Guy Pérennou. *Les vérificateurs orthographiques.* Actes du Colloque interdisciplinaire « Texte et Ordinateur ; les mutations du lire-écrire » Université Paris X Nanterre. pp 55-86. - Juin 1990.

(Pérennou 91) Guy Perennou, Chapitre 4 de (Haton 91) *Reconnaissance automatique de la parole.* DUNOD Informatique - 1991

(Pérennou 92) Guy Pérennou, D. Cotto, M. De Calmes. *Le projet BDLEX de base de données lexicales du français écrit et parlé.* Les actes des journées GRECO-PRC. Communication Homme Machine - Séminaire LEXIQUE - Toulouse - Janvier 1992.

(Pierrel 98) J.M. Pierrel - *Bilan des grandes orientations de recherche en traitement automatique du langage parlé en France* – Revue semestrielle de l'ATALA – Volume 38, n° 2, pp. 7-46 – Janvier 1998.

(Pollock 84) J. Pollock, A. Zamora *Automatic spelling correction in scientific and scholarly text CACM* – 1984.

(Pompidor 95) Pierre Pompidor, Jean-François Vergnaud. *Coopération et révision des agents d'un système coopératif gérant des bases de données. Application à la traduction automatique du chinois dans un but pédagogique.* Actes des Troisièmes Journées Francophones Intelligence Artificielle Distribuée & systèmes multi-agents. St Baldoph, Chambéry - 1995 -

(Ponton 96) Claude Ponton *Génération automatique de textes en langues naturelles. Essai de définition d'un système noyau.* Thése de doctorat en Informatique et Communication. Université Stendhal, Grenoble 3. - Novembre 1996.

(Rajman 95) Martin Rajman. *Approche probabiliste de l'analyse syntaxique. Traitements probabilistes et corpus* -Volume 36 - Revue semestrielle de l'ATALA - 1995 -

(Rouault 87) J. Rouault *Linguistique automatique* Edition Peter Lang - 1987.

(Rousseau 93) D. Rousseau, B. Moulin, G. Lapalme. *Modélisation des conversations basée sur une perspective de systèmes multi-agents.* Actes du colloque Informatique & Langue naturelle ILN.'93 - Nantes - Décembre 1993.

(Ruegg 89) Alan Ruegg. *Processus Stochastiques.* Presses Polytechniques Romandes - 1989.

(Sabah 88) Gérard Sabah. *L'intelligence artificielle et le langage, tome 1 : représentation des connaissances.* Edition HERMES – 1988.

(Sabah 89) Gérard Sabah. *L'intelligence artificielle et le langage, tome 2 : processus de compréhension.* Edition HERMES – 1989.

(Sabah 90) Gérard Sabah. *CARAMEL a computational model of natural language understanding using a parallel implementation.* E.C.A.I. – 1990.

(Saint-Dizier 93) Patrick SAINT-DIZIER. *Le traitement automatique de la langue écrite.* La recherche en informatique, Le courrier du CNRS N°80, Février 1993 .

(Sian 90) S. S. Sian. *Adaptation based on Cooperative Learning in Multi-Agents Systems.* Proceedings of the First European Wookshop on Modeling an

Autonomous Agent in a multi-Agent World. Elsevier Science Publishers (North Holland) - July 1990.

(Sta 95) Jean-David Sta. *Comportement Statistique des termes et Acquisition Terminologique à partir de Corpus.* Traitements probabilistes et corpus - Volume 36 - Revue semestrielle de l'ATALA - 1995 -

(Stefanini 93) Marie-Hélène Stefanini. *TALISMAN : une architecture multi-agents pour l'analyse du français écrit.* Thèse de doctorat - Université Pierre Mendès France - Grenoble - Janvier 1993.

(Stefanini 95) Marie-Hélène Stefanini, Yves Demazeau. *TALISMAN : a multi-agent system for natural language processing.* 12 th Brazilian symposium on artificial Intelligence. Caminas, Brazil – Octobre 1995.

(Stefanini 97) Marie-Hélène Stefanini, Karine Warren. *La syntaxe dans un univers multi-agents.* Acte de FRACTAL-97- Besançon - Décembre 1997

(Strube de Lima 89) Vera Lucia STRUBE DE LIMA. *Contribution à l'étude du traitement des erreurs au niveau lexico-syntaxique dans un texte écrit en français.* Thèse de l'Université Joseph Fourier, Grenoble 1, Mars 1990 .

(Tigli 94) J.Y. Tigli, M.C. Thomas. *La stratégie d'un système multi-agents pour le contrôle en temps réel.* Actes des Deuxièmes Journées Francophones Intelligence Artificielle Distribuée & systèmes multi-agents - Voiron - Mai 1994

(Tomasino 90) Isabelle Tomasino. *ODILE : Un Outil d'Intégration Extensible de Dictionnaires et de Lemmatiseurs.* Thèse CNAM, Grenoble - Décembre 1990 .

(Tomita 84b) Masaru Tomita. *LR parsers for natural language.* 10th Coling, Stanford, USA, pp 354 - 357.- July 1984.

(Tomita 84b) Masaru Tomita. *Disambiguating Grammatically Ambigyous Sentences By Asking.* 10th Coling, Stanford, USA, pp 476 - 480.- July 1984.

(Tomita 87) Masaru Tomita. *An efficient augmented Context free parsing algorithm.* AJCL pp 31,46. - 1987.

(Trouilhet 94) Sylvie Trouilhet. *Méthode d'acquisition des connaissances sociales pour l'organisation d'une société d'agents.* Actes des Deuxièmes Journées Francophones Intelligence Artificielle Distribuée & systèmes multi-agents - Voiron - Mai 1994

(Veronis 88) Jean Veronis. *Contribution à l'étude de l'erreur dans le dialogue homme machine en langage naturel.* Thèse de doctorat - Université de Droit, d'Economie d'Aix Marseille - Octobre 1988.

(Vivet 91) M.Vivet. *Expertise pédagogique et usage des tuteurs.* Actes des XIII journées francophones sur l'informatique. pp 135-143. Genève- Janvier 1991

(Vignal 94) Laurence Vignal. *La négociation : une utilisation constructive des conflits en résolution de problèmes.* Actes des Deuxièmes Journées Francophones Intelligence Artificielle Distribuée & systèmes multi-agents - Voiron - Mai 1994

(Wagner 74) C. Wagner, M. Fischer. *The string-to-string correction problem.* Journal of the A.C.M. Vol 21 n° 1 pp 168-173 - 1974.

(Warren 96) Karine Warren, Marie-Hélene Stefanini. *Modélisation et validation de protocoles de communication dans l'architecture* NLP+IA 96, Moncton, Canada, - Juin 1996.

(Warren 98) Karine Warren. *Gestion de conflits dans une architecture multi-agents d'analyse automatique de textes.* Thèse de l'université Stendhal, Grenoble 3.- Janvier 1998.

(Weischedel 83) Ralph. H. Weischedel. *Meta Rules as a Basic for Processing Ill-formed Input.* AJCL 9:3-4, , pp 161 – 177- 1983.

(Winograd 72) T Winograd. *Understanding natural language* – Academic Press, Edinburgh - 1977.

(Zweigenbaum 89) P. Zweigenbaum, Bruno Bachimont, Jacques Bouaud, *HELENE Compréhension de comptes rendus d'hospitalisation* – Deuxième école d'été sur le traitement des langues naturelles, Lanion, ENSAT – Juillet 1989.

11. ANNEXES

12. CATÉGORIES LEXICALES

Liste des catégories lexicales et de leurs abréviations

adv	"Adverbe"
subc	"substantif commun"
detp	"Déterminant-pronom"
det	"Déterminant"
subp	"Substantif propre"
adjq	"Adjectif qualificatif"
vide	""
infi	"Infinitif"
ppt	"Participe présent"
ppas	"Participe passé
verb	"Verbe"
xet "	Auxiliaire être"
xav	"Auxiliaire avoir" p
pnt	"Point" 15 -1 -1
dpnt	"Deux points"
virg	"Virgule"
coco	"Conjonction de coordination"
prep	"Préposition"
locp	"Locution préposition"
cocs	"Conjonction subordination"
locs	"Locution conjonctive"
padv	"Pronom adverbial y en"
prl	"Pronom relatif sauf "
prlc	"Pronom relatif conjonctif"
ne	"Négation ne"
pas	"2ème négation pas"
ce	"Pronom démonstratif"
pper	"Pronom personnel"
phra	"Phrase" "CL syntaxique"
vet	"Verbe être"
pnp	"Pronom non personnel"
loca	"Locution adverbiale"
ppf	"Pronom personnel fort"
adji	"Adjectif indéfini"
chf	"Chiffre"
date	"Date"
intj	"Interjection"
quan	"Quantification"
vav	"Verbe avoir"
de	"Préposition de"
à	"prep et à la (au)" "CL syntaxique"
cls	"SuperCl"

Remarque : les textes issus de l'oral ont été étudiés avec une liste légèrement différente.

Listes des variables et de leurs abréviations

sin	"Singulier"
plu	"Pluriel"
fem	"Féminin"
mas	"Masculin"
tre	"Troisième personne"
cod	"Complément d'objet direct"
ide	"Indéfini"
pat	"Partitif"
uno	"Première personne"
dos	"Deuxième personne"
pre	"Présent"
ind	"Indicatif"
fut	"Futur"
cdl	"Conditionnel"
imi	"Imparfait"
sub	"Subjonctif"
pas	"Passé simple"
imp	"Impératif"
suj	"Sujet"
dat	"Complément d'objet indirect"
cpr	"Complément prépositionnel"
ref	"Réfléchi"
cdn	"Complément de nom"

13. EXEMPLE DE TEXTE : MURIP

MURIP est un exemple de la dizaine de textes, tiré de l'oral pour lesquels nous disposons, outre les versions initiales, de versions corrigées, de versions avec analyses morphologiques complètes et de versions désambiguïsées manuellement (Frechet 92).

L'utilisation de ces textes ne visait pas une spécialisation dans le domaine de l'oral mais permettait d'avoir un réservoir de phrases entachées d'erreurs avec une syntaxe épouvantable. De plus, le coûteux travail de désambiguïsation étant fait, nous avons pu nous en servir

1. pour la mise au point des agents utilisant les modèles de Markov,
2. pour la définition du MLPS (lexique et grammaire) d'un rédacteur

*je m'appelle Murielle (h) et: je veux lancer le programme "PTS" phase 1 [h] (h) j'appuie donc sur les touches correspondantes "PTS" (h) et je lance avec "Go" (h) "return" [h] (h) je veux continuer (m) j'appuie sur e n'importe quelle touche je continue (h) et: je continue (h),, donc je prends la: la souris (h) et: j'aimerais sélectionner une [h] j'aimerais tout d'abord écouter (h) le signal qui a été enregistré (h), e: donc ja e sélectionne "processing" (h) et: je veux écouter tout donc "listen all" /,,/ e,, (h) je veux sélectionner une partie du signal (h) mais je trouve pas les touches, (+),, (h) (m),, donc je dépla/ (h) je déplace le curseur pour sélectionner une partie du signal, et le curseur gauche je le ramène oh oh, bon [h] le curseur ne veut pas avancer, (m) (h) (+) bon tant pis [h], (h) je ramène le curseur droit, et: je veux zoomer la sélection [h], (m) (h),, e la sélection que j'ai fait j'aimerais en voir un spectrogramme, ok on peut lancer, je, ça marche pas, (+) (h) bon [h], on a/ ah voilà,, ,, (+) c'est terminé j(e) vais: changer les couleurs du signal, j(e) demande la "palette", et: cette palette il faudrait la remonter (h) parce qu'elle me: j cache une partie du spec/ oh oh, là (h) j'aimerais avoir que les couleurs vertes sur l'écran [h] donc j'enlève toutes les bleues, je clique sur les bleues, i(l) m(e) reste les vertes, main(te)nant j'aimerais l(e) contraire avoir toutes les bleues (h) voilà celle-là il est en moins, avoir toutes les bleues mais pas les vertes donc i(l) faut remettre les bleues, et enlever les vertes, et on va même enlever l(e) gris, bon alors on peut l'avoir à nouveau comme il était au début (h) donc il faut remettre les vertes (h) et on reclique sur les vertes (+) bon on a plus besoin de la palette donc on peut l'enlever,, (h), on, on réactive la première fenêtre, et on resélectionne un petit morceau d/ (h), (mm) le curseur droit voilà, jusque-là et le curseur gauche est-ce qu'i(l) va s(e) décider cette fois? merveilleux, et voilà (m) (h) on ref/ on le: on réagrandit ça (h) donc je zoome la sélection (h), (h) et: on va mesurer c(e) qu'on a fait ** donc j'ai un curseur au milieu (h) que je déplace [h] et on: la fenêtre m'indique les mesures, donc je mesure seulement une partie, voilà,*

c'est bon, (h) j(e) laisse tomber (h), j'ai plus b(e)soin d(e) mesurer i(l) peut partir, "file out" ah non non non non non, (+) je "quit", (h) ben on va redessiner juste la: (h) ce que je viens d(e) sélectionner en sonagramme donc je: rappelle "processing" (h) je sélectionne le "sonagramme" (h) e: les paramètres (h) bon ben i(ls) sont bons on y va "OK" (h) le sonagramme se dessine bien lentement (h),, ,, (h) je resélectionne la: fenêtre du signal de parole (h) et j'aimerais la l'en déplacer vers le haut (h) donc tu remontes là, voilà (h) et: (h) j(e) voudrais le signal initial, comment est-ce qu'on l'appelle celui-là, on peut pas l'appeler (+) (h) ah, "control" (m) "conflict" ah ben on va essayer "conflict" non, ah (+) j'ai quitté sans faire exprès ah c'est pas grave c'est fini (=)

etc.

14. INTERFACES

14.1 ACTIVATION DES AGENTS

14.1.1 ACTIVATION SANS ARGUMENT

L'appel d'un agent sans argument provoque l'affichage de quelques lignes d'aide rappelant sommairement la signification et l'utilisation correcte des arguments par rapport aux fonctionnalités attendues.

14.1.2 ACTIVATION AVEC ARGUMENT(S)

Les agents peuvent être activés selon divers modes de communication : socket, pipe et pour certains offrent la possibilité d'un accès direct à un menu (paramétrage) ou à un fonctionnement en ligne de commande.

Dans la ligne de commande, le premier argument représente le mode de communication selectionné :

Argument 1		**Argument 2**	**Arguments suivants 3 ...**
m	Appel du menu d'interface		
s	Communication par socket	N° du port du socket d'écoute	
t	Communication par tube (pipe)		
l	Fonctionnement en ligne de commande	Fonction attendue de l'agent	Les données (paramètres) nécessaires à la fonction.

Prenons à titre d'exemple commenté quelques fonctionnements de l'agent pscm :

Activation de l'agent	
pscm	Affichage de quelques lignes d'indications sur l'usage de l'agent
pscm m	Appel du **m**enu (voir § C.2 sur l'agent pscm ; ce menu permet diverses fonctionnalités de contrôle).
pscm s 2000	Activation de l'agent pscm avec un **s**ocket d'écoute sur le port 2000.

pscm l P Einstein 0.450 0.500 det adji verb adv	Activation en ligne de commande Demande pour l'utilisateur Einstein d'une **P**roposition de catégories morphologiques à l'agent avec réponse sur le terminal: 0.450 seuil de probabilité pour l'ordre 1 0.500 seuil de probabilité pour l'ordre 2 det adji contextegauche verb adv contexte droit.
pscm l A Einstein det+adji+subc+verb+ adv	Activation en **l**igne de commande Demande d'**A**pprentissage de l'échantillon det adji subc verb adv pour l'utilisateur Einstein.
pscm l 3 Einstein 0	Initialisation à zéro des matrices de l'utilisateur Einstein.

14.2 AGENT PSCM

• **Mode 1 interface utilisateur de gestion du module** : cette interface sommaire présente un menu général permettant diverses opérations sur le paramétrage du module telles que l'inscription d'un nouveau client, la ré-initialisation à zéro des matrices de transitions, l'affichage des catégories morphologiques etc.
Le mode 1 est choisi en activant l'agent sans argument :

pscm

```
AGENT PSCM : menu des interventions directes


     Affichage de la liste des catégories morphologiques ........ L
     Création d'un fichier explicite de la
            matrice de transition directe d'ordre 1 ............. U
            matrice de transition directe d'ordre 2 ............. D
            matrice de transition inverse d'ordre 1 ............. V
            matrice de transition inverse d'ordre 2 ............. E
     Mises à jour des matrices de transitions par ajout ......... M
     Initialisation à zéro des matrices de transition ........... I
     Test et trace à l'écran des propositions ................... T
     Quitter .................................................... Q

Votre choix ............................ (L/U/D/V/E/M/I/T/Q ?) :
```

14.3 AGENT PILCHART

L'analyse proprement dite et le résultat de l'analyse se traduisent par l'apparition de différentes fenêtres (trace dans la fenêtre LISTENER, temps, résultats dans une cascade de fenêtres présentant les différents arbres solutions).

Le deuxième écran correspond à une re disposition des fenêtres des deux arbres solutions de cette phrase ambiguë.

Rappelons que l'objectif de PILCHART est d'être un outil pratique de génie linguistique : tester des grammaires, tester des stratégies. Les résultats devaient apparaître sous une forme graphique "parlante" et donc sous forme de dessin d'arbres. L'interface simple

était absolument nécessaire pour permettre par exemple à un linguiste non informaticien d'utiliser le logiciel.

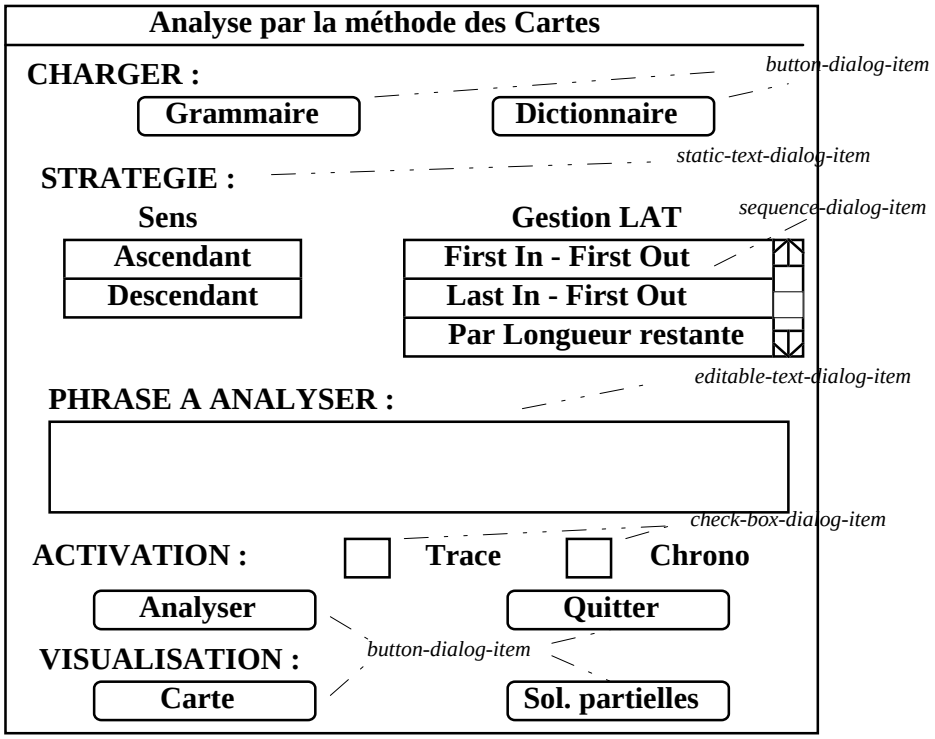


figure C.1 : Présentation du menu général

14.4 L'AGENT TJRP

```
AGENT TJRP : menu des interventions directes


    Affichage de la liste des couples de mots remplacé/remplacant.. L
    Ajout d'un couple de mots ..................................... A
    Suppression d'un couple de mots ............................... S
    Initialisation a zero du fichier des couples .................. I
    trace a l'ecran du Remplacement ............................... R
    Quitter ....................................................... Q

Votre choix ........................... (L/A/S/I/R/Q ?) : q
```

15. TRACE D'EXÉCUTION DE CORRECTION DES ERREURS

L'exemple choisi

le beau chevaux marron racée et les juments courent furieux et affamés

conduit à examiner diverses solutions concurrentes avant de trancher.

15.1 DÉTERMINATION DES UNIFICATIONS PERTINENTES PAR L'AGENT PAVA

En partant du résultat de l'analyse syntaxique de la phrase complète et donnée ici sous la forme d'une liste du type LISP:

P(gn(gn(detp(le,mas,sin)adjq(beau,mas,sin)subc(chevaux,mas,plu)adjq(marron,mas,sin)adjq(racée,fem,sin))coco(et)gn(detp(les,mas,fem,plu)subc(juments,fem,plu)))gv(verb(courent,plu)adv(adjq(furieux,mas,sin,plu)coco(et)ppas(affamés,mas,plu)))))

l'agent PAVA commence par découper la phrase en structures. Pour ce type de décomposition, les liens entre les structures sont mémorisés sous la forme du libellé de l'arbre lexicographique des chemins :

.1/p
.1.1/gn
<=(gn(detp(le,mas,sin)adjq(beau,mas,sin)subc(chevaux,mas,plu)adjq(marron,mas,fem,sin,plu)adjq(racée,fem,sin))coco(et)gn(detp(les,mas,fem,plu)subc(juments,fem,plu)))
.1.1.1/gn
<=(detp(le,mas,sin)adjq(beau,mas,sin)subc(chevaux,mas,plu)adjq(marron,mas,fem,sin,plu)adjq(racée,fem,sin))
.1.1.1.1/detp <=(le,mas,sin)
.1.1.1.2/adjq <=(beau,mas,sin)
.1.1.1.3/subc <=(chevaux,mas,plu)
.1.1.1.4/adjq <=(marron,mas,fem,sin,plu)
.1.1.1.5/adjq <=(racée,fem,sin)
.1.1.2/coco <=(et)
.1.1.3/gn <=(detp(les,mas,fem,plu)subc(juments,fem,plu))
.1.1.3.1/detp <=(les,mas,fem,plu)
.1.1.3.2/subc <=(juments,fem,plu)
.**1.2/gv**
<=(verb(courent,plu)adv(adjq(furieux,mas,sin,plu)coco(et)ppas(affamés,mas,plu)))
.**1.2.1/verb** <=(courent,plu)
.**1.2.2/adv** <=(adjq(furieux,mas,sin,plu)coco(et)ppas(affamés,mas,plu))
.**1.2.2.1/adjq** <=(furieux,mas,sin,plu)
.1.2.2.2/coco <=(et)
.1.2.2.3/ppas <=(affamés,mas,plu)

En partant des structures terminales et en remontant dans l'arborescence, le module PAVA va passer au module DCFA :

- Le "paquet" de structures à traiter;
- La stratégie à utiliser.

Pour traiter notre exemple, parmi les stratégies possibles, nous supposerons que la stratégie choisie est celle des seuils :

Stratégie des seuils

Par exemple, pour un seuil pour le genre de 0.5,

si le genre calculé de la structure est de 0,6 (supérieur au égal au seuil),

le module tranchera,

décidera du genre masculin pour la structure,

demandera la génération morphologique des formes à corriger s'il en existe.

si le genre calculé est de -0.6 (inférieur ou égal à l'opposé du seuil),

le module tranchera

décidera du genre féminin pour la structure.

demandera la génération morphologique des formes à corriger s'il en existe

si le genre calculé est compris strictement entre -0.5 et +0.5, le module DCFA s'abstiendra de toute conclusion et en informera le module PAVA.

Si la valeur du seuil est fixée à 1, le module DCFA va alors se contenter de reconnaître les structures sans fautes et de signaler les structures posant problèmes.

15.2 TRAVAIL DE L'AGENT DCFA

Le module d'unification PAVA va tout d'abord demander l'examen des structures dont les composants sont des structures terminales. L'ordre lexicographique sur les chemins étant une relation d'ordre, les unifications possibles vont être traitées par ordre décroissant en partant du niveau le plus bas.

Nom de la structure		Variables Morpho.	genre	nombre	nbmt	nbmg	nbmn
adv	*furieux*	mas, sin, plu	1	0	1	1	0
	et	Coco	0	0	1	0	0
	affamés	mas, plu	1	1	1	1	1
Qualité :			1	1	3	2	1
Adv			1/1/3/2/1/adv(*furieux et affamés*,mas, plu))				

Nom de la structure		Variables Morpho.	genre	nombre	nbmt	nbmg	nbmn
det	*les*	fem, mas, plu	0	1	1	0	1
subc	*juments*	fem, plu	-1	1	1	1	1
Qualité :			-1	1	2	1	2
Gn			-1/1/2/1/2/gn(*les juments*, fem plu)				

Nom de la structure			genre	nombre	nbmt	nbmg	nbmn

det	*le*	mas, sin	1	-1	1	1	1
adjq	*beau*	mas, sin	1	-1	1	1	1
subc	*chevaux*	mas, plu	1	1	1	1	1
adjq	*marron*	mas, fem, sin	0	0	1	0	0
adjq	*racée*	fem, sin	-1	-1	1	1	1
Qualité :			0.5	-0.5	5	4	4
Gn			0.5/-0.5/5/4/4/gn(l*e beau cheval marron racé*, mas, sin)				

L'agent DCFA en appliquant la stratégie demandée par l'agent PAVA va demander les générations morphologiques de *racée* au *mas, sin* et de *chevaux* au *mas, sin*
ce qui va générer la forme correcte *le beau cheval marron racé* en remplacement de *le beau chevaux marron racée*.

Nom de la structure		Variables Morpho.	genre	nombre	nbmt	nbmg	nbmn
verb	*courent*	Plu	0	1	1	0	1
adv	*furieux et affamés*	mas, plu	1	1	3	2	1
Qualité :			1	1	4	2	2
Gv			-1/1/4/2/2/gv(courent furieux et affamés, mas, plu)				

Nom de la structure		Variables Morpho.	genre	nombre	nbmt	nbmg	nbmn
gn	*le beau cheval marron racé*	mas, sin, plu	0.5	-0.5	5	4	4
coco	*et*	Coco	0	0	1	0	0
gn	*les juments*	fem, plu	-1	1	2	1	2
Qualité :			1	1	3	2	1
Gn			1/1/8/5/6/gn(*le beau cheval marron racé et les juments*, mas, plu)				

Nom de la structure		Variables Morpho.	genre	nombre	nbmt	nbmg	nbmn
gn	*le beau cheval marron racé et les juments*	mas, plu	1	1	8	5	6
gv	*courent furieux et affamés*	mas, plu	1	1	4	2	2
Qualité :			1	1	12	7	8
P			*le beau cheval marron racé et les juments courent furieux et affamés*				

1	**2**
?/?/?/?/?/p () ?/?/?/?/?/gn () ?/?/?/?/?/gn () 1/-1/1/1/1/detp(le,mas,sin) 1/-1/1/1/1/adjq(beau,mas,sin) 1/1/1/1/1/subc(chevaux,mas,plu) 0/0/1/0/0/adjq(marron,mas,fem,sin,plu) -1/-1/1/1/1/adjq(racée,fem,sin) 0/0/1/0/0/coco(et) ?/?/?/?/?/gn () 0/1/1/0/1/detp(les,mas,fem,plu) -1/1/1/1/1/subc(juments,fem,plu) ?/?/?/?/?/gv () 0/1/1/0/1/verb(courent,plu) 1.000/1.000/3/2/1/adv(furieux et affamés ,mas ,plu) 1/0/1/1/0/adjq(furieux,mas,sin,plu) 0/0/1/0/0/coco(et) 1/1/1/1/1/ppas(affamés,mas,plu)	?/?/?/?/?/p () ?/?/?/?/?/gn () ?/?/?/?/?/gn () 1/-1/1/1/1/detp(le,mas,sin) 1/-1/1/1/1/adjq(beau,mas,sin) 1/1/1/1/1/subc(chevaux,mas,plu) 0/0/1/0/0/adjq(marron,mas,fem,sin,plu) -1/-1/1/1/1/adjq(racée,fem,sin) 0/0/1/0/0/coco(et) -1.000/1.000/2/1/2/gn(les juments ,fem ,plu) 0/1/1/0/1/detp(les,mas,fem,plu) -1/1/1/1/1/subc(juments,fem,plu) ?/?/?/?/?/gv () 0/1/1/0/1/verb(courent,plu) 1.000/1.000/3/2/1/adv(furieux et affamés ,mas ,plu) 1/0/1/1/0/adjq(furieux,mas,sin,plu) 0/0/1/0/0/coco(et) 1/1/1/1/1/ppas(affamés,mas,plu)
3	**4**
?/?/?/?/?/p () ?/?/?/?/?/gn () 0.500/-0.500/5/4/4/gn(le beau cheval marron racé ,mas ,sin) 1/-1/1/1/1/detp(le,mas,sin) 1/-1/1/1/1/adjq(beau,mas,sin) 1/1/1/1/1/subc(chevaux,mas,plu) 0/0/1/0/0/adjq(marron,mas,fem,sin,plu) -1/-1/1/1/1/adjq(racée,fem,sin) 0/0/1/0/0/coco(et) -1.000/1.000/2/1/2/gn(les juments ,fem ,plu) 0/1/1/0/1/detp(les,mas,fem,plu) -1/1/1/1/1/subc(juments,fem,plu) ?/?/?/?/?/gv () 0/1/1/0/1/verb(courent,plu) 1.000/1.000/3/2/1/adv(furieux et affamés ,mas ,plu) 1/0/1/1/0/adjq(furieux,mas,sin,plu) 0/0/1/0/0/coco(et) 1/1/1/1/1/ppas(affamés,mas,plu)	?/?/?/?/?/p () ?/?/?/?/?/gn () 0.500/-0.500/5/4/4/gn(le beau cheval marron racé ,mas ,sin) 1/-1/1/1/1/detp(le,mas,sin) 1/-1/1/1/1/adjq(beau,mas,sin) 1/1/1/1/1/subc(chevaux,mas,plu) 0/01/1/0/0/adjq(marron,mas,fem,sin,plu) -1/-1/1/1/1/adjq(racée,fem,sin) 0/0/1/0/0/coco(et) -1.000/1.000/2/1/2/gn(les juments ,fem ,plu) 0/1/1/0/1/detp(les,mas,fem,plu) -1/1/1/1/1/subc(juments,fem,plu) 1.000/1.000/4/2/2/gv(courent furieux et affamés ,mas ,plu) 0/1/1/0/1/verb(courent,plu) 1.000/1.000/3/2/1/adv(furieux et affamés ,mas ,plu) 1/0/1/1/0/adjq(furieux,mas,sin,plu) 0/0/1/0/0/coco(et) 1/1/1/1/1/ppas(affamés,mas,plu)

5	**6**
?/?/?/?/?/p ()	1.000/1.000/12/7/8/p(le beau cheval marron racé et les juments courent furieux et affamés ,mas ,plu)
1.000/1.000/8/5/6/gn(le beau cheval marron racé et les juments ,mas ,plu)	1.000/1.000/8/5/6/gn(le beau cheval marron racé et les juments ,mas ,plu)
0.500/-0.500/5/4/4/gn(le beau cheval marron racé ,mas ,sin)	0.500/-0.500/5/4/4/gn(le beau cheval marron racé ,mas ,sin)
1/-1/1/1/1/detp(le,mas,sin)	1/-1/1/1/1/detp(le,mas,sin)
1/-1/1/1/1/adjq(beau,mas,sin)	1/-1/1/1/1/adjq(beau,mas,sin)
1/1/1/1/1/subc(chevaux,mas,plu)	1/1/1/1/1/subc(chevaux,mas,plu)
0/0/1/0/0/adjq(marron,mas,fem,sin,plu)	0/0/1/0/0/adjq(marron,mas,fem,sin,plu)
-1/-1/1/1/1/adjq(racée,fem,sin)	-1/-1/1/1/1/adjq(racée,fem,sin)
0/0/1/0/0/coco(et)	0/0/1/0/0/coco(et)
-1.000/1.000/2/1/2/gn(les juments ,fem ,plu)	-1.000/1.000/2/1/2/gn(les juments ,fem ,plu)
0/1/1/0/1/detp(les,mas,fem,plu)	0/1/1/0/1/detp(les,mas,fem,plu)
-1/1/1/1/1/subc(juments,fem,plu)	-1/1/1/1/1/subc(juments,fem,plu)
1.000/1.000/4/2/2/gv(courent furieux et affamés ,mas ,plu)	1.000/1.000/4/2/2/gv(courent furieux et affamés ,mas ,plu)
0/1/1/0/1/verb(courent,plu)	0/1/1/0/1/verb(courent,plu)
1.000/1.000/3/2/1/adv(furieux et affamés ,mas ,plu)	1.000/1.000/3/2/1/adv(furieux et affamés ,mas ,plu)
1/0/1/1/0/adjq(furieux,mas,sin,plu)	1/0/1/1/0/adjq(furieux,mas,sin,plu)
0/0/1/0/0/coco(et)	0/0/1/0/0/coco(et)
1/1/1/1/1/ppas(affamés,mas,plu)	1/1/1/1/1/ppas(affamés,mas,plu)

16. TRACES D'EXÉCUTION DES AGENTS BRUNO, CELINE, ALMLPS ET ASMLPS

Les agents possèdent en mode trace la possibilité de les activer avec l'option *: Travail à une passe*. Sous cette option, ils effectuent un travail atomique sur le tableau noir. Le superviseur CELINE va afficher le contenu du tableau noir après chaque action.

Remarques :

1. les traces sont des traces « brutes » pour informaticiens.
2. Les indices varient à partir de zéro

Agent BRUNO

```
 Quitter ..................................... Q
 Votre choix L/T/A/Q .....................? : S


Texte initial :
Les chiens aboient. La caravane passent.

Decoupage du corpus en structure :
Phrase 0 :
M=Les=S= =M=chiens=S= =M=aboient=S=.=
Phrase 1 :
M==S= =M=La=S= =M=caravane=S= =M=passent=S=.=
 AGENT BRUNO

 Liste des ponctuations ..................... L
 Test de la mise au format.....................F
 Test du decoupage en structure ..............S
 Activation de l'agent.........................T
 Quitter ..................................... Q
 Votre choix L/T/A/Q .....................? :
```

L'agent a découpé le texte de deux phrases. Le vecteur des séparateurs peut contenir des espaces ou ponctuations mais aussi le format (Word ou XML) ainsi que les informations lexico-syntaxiques que nous nous mémorisons par mot.

Agents CELINE, MORPHO et ALMLPS

```
 AGENT CELINE

 Visualisation du tableau noir.............. V
 Activation de l'agent.........................T
 Quitter ..................................... Q
 Votre choix L/T/A/Q .....................? : v


 CONTENU DU TABLEAU NOIR :
0) Phrase 0 Mot 0 Ambi 0 : le detp s m       Valid : V V X X X X X X X X
1) Phrase 0 Mot 0 Ambi 0 : le detp s n       Valid : V V X X X X X X X X
2) Phrase 0 Mot 1 Ambi 0 : joli adjq s m       Valid : V V X X X X X X X X
3) Phrase 0 Mot 2 Ambi 0 : cheval subc s m       Valid : V V X X X X X X X X
4) Phrase 0 Mot 3 Ambi 0 : zebul            Valid : I I X X X X X X X X
5) Phrase 0 Mot 4 Ambi 1 : court verb          Valid : V V X X X X X X X X
6) Phrase 0 Mot 4 Ambi 2 : court adv i i       Valid : V V X X X X X X X X
7) Phrase 0 Mot 4 Ambi 3 : court adjq s m       Valid : V V X X X X X X X X
8) Phrase 0 Mot 5 Ambi 0 : vite adv d d       Valid : V X X X X X X X X X

 AGENT CELINE

 Visualisation du tableau noir.............. V
 Activation de l'agent.........................T
 Quitter ..................................... Q
 Votre choix L/T/A/Q .....................? :
```

Les agents MORPHO et ALMLPS ont travaillé sur le tableau noir. Les catégories morphologiques et les variables morphologiques sont apparentes. Les deux premières marques de validité ont été affectées.

Remarque : certaines variables (i, d) ne correspondent volontairement à rien car nous testions aussi les tables de correspondances inter traits agents (extérieur / CELINE)

AGENT ASMLPS

Trace de la mini-grammaire utilisée :

```
Regle 1 : GV --> verb
Regle 2 : GV --> verb adv
Regle 3 : GN --> det subc
Regle 4 : GN --> det adjq subc
Regle 5 : GN --> det subc adjq
Regle 6 : GN --> GN coco GN
Regle 7 : P --> GN GV
```

Là encore, les tables de correspondances sont testées (catégories *det* et *dept*).

```
 CONTENU DU TABLEAU NOIR :
0) Phrase 0 Mot 0 Ambi 0 : le detp s m       Valid : V V X X X X X X X X
1) Phrase 0 Mot 0 Ambi 0 : le detp s n       Valid : V V X X X X X X X X
2) Phrase 0 Mot 1 Ambi 0 : joli adjq s m       Valid : V V X X X X X X X X
3) Phrase 0 Mot 2 Ambi 0 : cheval subc s m       Valid : V V X X X X X X X X
4) Phrase 0 Mot 3 Ambi 0 : zebul           Valid : I I X X X X X X X X
5) Phrase 0 Mot 4 Ambi 1 : court verb          Valid : V V X X X X X X X X
           Struct 0 : (0(5  ))
           Struct 1 : (1(5  8  ))
6) Phrase 0 Mot 4 Ambi 2 : court adv i i       Valid : V V X X X X X X X X
7) Phrase 0 Mot 4 Ambi 3 : court adjq s m       Valid : V V X X X X X X X X
8) Phrase 0 Mot 5 Ambi 0 : vite adv d d      Valid : V X X X X X X X X X

 AGENT CELINE

 Visualisation du tableau noir..... V
 Activation de l'agent.....T
 Quitter ..... Q
 Votre choix L/T/A/Q .....? :
```

L'analyseur syntaxique vient de trouver deux structures terminales : un groupe verbal (*court*) formé d'un verbe (règle 1) ou un groupe verbal formé d'un verbe (*court*) et d'un adverbe (*vite*) (règle 2).

RÉSUMÉ

Cette thèse aborde la spécification et la réalisation de CELINE, outil de correction des erreurs basé sur une architecture multi-agents à deux niveaux :
Un système lourd, renfermant l'ensemble du savoir linguistique (multi-domaines par rapport à l'univers du discours), et générateur de systèmes individualisés. Les agents peuvent être considérés comme imparfaits ou partiellement inadaptés. Ils sont mis en concurrence par domaine d'expertise.
Un système léger implanté sur le site du rédacteur, système construit par apprentissage par le système central à partir des travaux de ce rédacteur.

La spécification du système se construit tout au long des chapitres.
La problématique de la correction des erreurs et la finalité de la conception d'un système de correction le plus automatique possible, avec des prises de décision à faible granularité reposant sur des critères multi-niveaux, nous entraînent vers un besoin de coopération justifiant une réalisation multi-agents.
Une taxinomie des erreurs et des rappels sur l'analyse linguistique nous permet d'établir un début de structure du tableau noir du système. Nous consolidons nos choix par une comparaison du système attendu avec quelques prototypes du domaine. Nous examinons ensuite les comportements sociaux de deux agents chargés de définir l'un un modèle linguistique partiel suffisant du rédacteur et l'autre un sous-ensemble pertinent du système global. Nous découvrons alors le modèle de communication des agents et complétons notre structure de données par les marques de validité.
La méthode des structures permet une quantification, incluse dans le tableau noir, de la correction des fautes d'accords.
Après une approche des systèmes multi-agents, nous présentons une synthèse de l'architecture de CELINE et du fonctionnement des pilotes et de quelques agents.

Un bilan rapide, précèdera en conclusion, une mise en situation du système proposé dans le cadre des industries de la langue et dans un environnement réseau du type Internet.

DISCIPLINE

INFORMATIQUE ; option SYSTÈMES ET COMMUNICATIONS.

MOTS-CLEFS

TRAITEMENT AUTOMATIQUE DES LANGUES NATURELLES
CORRECTEUR LEXICO-SYNTAXIQUE
VÉRIFICATION DES ACCORDS EN NOMBRE ET EN GENRE
INDUSTRIES DE LA LANGUE NATURELLE

INTELLIGENCE ARTIFICIELLE DISTRIBUÉE
SYSTÈMES MULTI-AGENTS
TABLEAUX NOIRS PARALLÈLES

LABORATOIRE CLIPS - Institut IMAG -
BP 53 - Grenoble Cedex 09- F 38041