UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN MATHÉMATIQUES ET INFORMATIQUE
APPLIQUÉES

PAR

BOUBAKER HAMROUNI

DÉSAMBIGUÏSATION DANS LA TYPIFICATION DE DONNÉES TEXTUELLES À DES FINS D'ANALYSES CATÉGORIELLES

MARS 2008

Sommaire

Ce mémoire s'inscrit dans le cadre des méthodes de traitement informatique du langage naturel. L'approche proposée se base sur la grammaire catégorielle qui permet sommairement d'appliquer des types catégoriels aux différents termes d'une phrase donnée en utilisant des règles de production précédemment fixées. Dans les applications réelles, une multitude de types catégoriels peuvent être assignés à une unité lexicale chose qui rend une analyse syntaxique trop complexe pour être efficace. A cet effet, il est d'une importance capitale de trouver une approche efficace pour limiter l'assignation des types catégoriels. Ceci est le principal objectif de notre travail qui consiste donc à concevoir et développer un outil permettant de trouver une approche efficace d'assignation des types catégoriels. Le travail réalisé dans ce mémoire a montré qu'une approche bayésienne permet d'offrir une amélioration substantielle quant à l'affectation des types catégoriels. Des tests ont été effectués sur un corpus composé de phrases benchmarks. Ils ont montré l'utilité de l'approche proposée étant donnée qu'elle présente un degré de précision assez élevé.

Dédicaces

Ce travail est dédié à mes Maîtres, mes parents et ma famille.

Qu'ils trouvent ici le témoignage de ma gratitude et de mon respect.

Remerciements

C'est avec un grand plaisir, que j'exprime ici ma reconnaissance à tous ceux qui m'ont apporté leur aide scientifique, matérielle ou morale pour mener à terme ce travail.

Le professeur Ismaïl Biskri du Département de Mathématiques et d'Informatique, qui m'a aidé dans ce travail et en a rendu possible la réalisation. J'ai particulièrement apprécié ses qualités humaines indéniables, sa disponibilité et ses conseils de bon aloi. Qu'il trouve ici l'expression de ma gratitude.

Les professeurs Louis Houde et François Meunier qui ont accepté d'être membres du jury. Ma profonde gratitude pour ces deux professeurs dont les évaluations m'ont été précieuses.

Je remercie tous les membres de département Mathématiques et Informatique pour m'avoir accueilli et fait bénéficier de leurs expériences.

Mes remerciements vont aussi à tous ceux qui, non cités ici, ont participé de près ou de loin par le savoir ou par les encouragements à l'élaboration de ce travail.

Toute ma reconnaissance et ma gratitude s'adressent particulièrement à mes parents qui ont contribué par leur amour et leur affection à l'aboutissement de ce travail et dont le soutien ne m'a jamais fait défaut. Une pensée particulière à mes frères et ma sœur pour leur soutien.

Introduction générale

Avec l'inépuisable croissance d'applications nécessitant un traitement efficace du langage naturel, il devient primordial de pouvoir identifier précisément les termes dans un domaine ou langage donné. À cette fin, deux types d'approche existent jusqu'à maintenant, soit une approche basée sur des techniques probabilistes, applicables à de vastes corpus et ce, indépendamment de la langue, soit une approche linguistique, formalisant les mécanismes linguistiques à l'aide d'un traitement symbolique, dépendant de la langue et sensible à certains néologismes.

Conscients des mérites et des limitations de chaque type d'approche, nous proposons l'emploi d'une autre méthode. L'approche catégorielle substitue un calcul différentiel à une analyse linguistique. Cette substitution passe par l'assignation d'un type catégoriel à chaque unité lexicale dans un premier temps, puis par une simplification des types catégoriels au noyau de règles catégorielles dans un deuxième temps.

La problématique : pour chaque entrée lexicale d'un dictionnaire catégoriel, il est possible de retrouver plusieurs types catégoriels. Comment arriver à sélectionner le bon type pour chaque unité lexicale étant donné l'énoncé à analyser ?

Les résultats de nos recherches, synthétisés en cinq chapitres dans ce mémoire, avancent une réponse à cette question.

Le premier chapitre aborde le cadre formel de la grammaire catégorielle ainsi que l'intérêt de cette dernière. Il définit également d'une manière détaillée la grammaire catégorielle combinatoire applicative.

Le deuxième chapitre traite de la théorie des graphes. Cette structure, càd, les graphes, est très utile pour caractériser les chemins formés par une séquence de types associés à des termes donnés.

Le troisième chapitre introduit la théorie des chaînes de Markov. Cette notion fondamentale sera utilisée afin de caractériser les liens entre les différents termes d'une phrase donnée. Ceci permettra d'associer une sémantique à la phrase.

Le quatrième chapitre présente les réseaux bayésiens. Ces derniers sont utiles pour caractériser la dépendance conditionnelle entre les probabilités d'apparition des types associés aux termes.

Le cinquième et dernier chapitre permet de décrire le prototype que nous avons développé. Ce prototype reflète les fondements théoriques décrits dans les chapitres précédents. Les tests ont été effectués sur un corpus formé par des phrases benchmark.

Le mémoire se conclut sur un résumé de l'ensemble de nos travaux et présente quelques perspectives futures de recherche.

Chapitre 1 : Grammaire catégorielle

1. Introduction

Le présent chapitre traite de la grammaire catégorielle, tout particulièrement intéressante pour nos travaux en raison de ses propriétés d'apprentissage. La grammaire catégorielle est parmi les plus anciens formalismes grammaticaux lexicalisés, un formalisme logique strict certes mais simple et efficace dans la représentation des phrases, d'où l'intérêt de l'utiliser dans le cadre d'un traitement informatisé. Les expressions grammaticales de la grammaire catégorielle se distinguent par un type syntactique les identifiant en tant que fonction ayant des arguments d'un type donné et ayant des résultats d'un autre type, ou bien, comme un argument de type donné. Bien qu'étroitement liés au type sémantique de l'expression linguistique, ces types, également appelés catégories, diffèrent des types sémantiques au niveau des informations spécifiques à un langage particulier [1].

Lors des premières apparitions de ce qui forme maintenant la grammaire catégorielle, les experts qualifièrent rapidement ces grammaires de « contexte libre » et faiblement équivalentes aux grammaires contexte libre à structure d'expression (noté dans la suite, GCLSE). L'établissement précoce de cette faible équivalence entre la grammaire catégorielle et les GCLSE a quelque peu écarté la grammaire catégorielle de la carte au cours des années 60. L'intérêt des syntacticiens envers la grammaire catégorielle vînt donc plus tard, à la fin années 70, pour se poursuivre dans les années 80. Les travaux de R. Montague au milieu des années 70 [2], ainsi que les travaux de ses successeurs, a suscité un gain d'intérêt notable grâce au développement accompli dans la formalisation d'une sémantique guidée par le type pour le langage naturel. Cela a rendu attrayant la transparence de la grammaire catégorielle au niveau du type syntaxique (ou sémantique).

Des opérations fonctionnelles sur des catégories adjacentes telles que la composition fonctionnelle et le « wrap » vinrent s'ajouter au noyau de la grammaire catégorielle, ce qui mena au rétablissement de l'intérêt dans l'alternative basée sur les types logiques issus des travaux de Lambek, fin années 50 [3].

Depuis ce temps, un trait assez marqué existe toujours entre les approches combinatoires et les approches basées sur les types logiques. D'une part, la grammaire catégorielle combinatoire s'est

concentrée à maintenir la puissance expressive au niveau « modérément sensible au contexte » identifié par Joshi et al. [4], et a été impliquée dans des problématiques d'explication linguistique et d'informatique linguistique pratique. D'autre part, les grammaires à base de types logiques ont été davantage concernées par les problématiques théoriques et les relations à la logique, la preuve des théorèmes et à la complexité.

Dans ce chapitre, nous allons nous concentrer principalement sur une présentation détaillée de la grammaire catégorielle. Il est important de noter qu'un état de l'art des différentes théories grammaticales se trouve dans [5].

2. Cadre formel des grammaires catégorielles

Les grammaires catégorielles sont définies comme suit :

Définition :

Les grammaires catégorielles sont des grammaires *totalement lexicalisées* : de manière simplifiée, cela signifie qu'une grammaire catégorielle est entièrement définie par son vocabulaire. Celui-ci est constitué de l'ensemble des terminaux (les mots), chaque terminal étant associé à une ou plusieurs *catégories*.

A cet effet, nous avons :

- Deux catégories de base à savoir S (pour « Sentence ») et N (pour « Noun »).
- Deux règles de formation des catégories complexes :
  - Les catégories de base sont des catégories.
  - Si X et Y sont des catégories alors X/Y (respectivement X\Y) sont des catégories. X/Y est la catégorie d'un opérateur ayant comme argument un opérande de type Y positionné à droite. X\Y est la catégorie d'un opérateur ayant comme argument un opérande de type X positionné à gauche.

Les dérivations sont effectuées à l'aide de deux règles logiques indépendantes de la grammaire. Ces règles sont les suivantes

Définition :

- Règle 1 : α, B / A, A, $\beta \rightarrow \alpha$, B, β.

- Règle 2 : α, A, A \ B, $\beta \rightarrow \alpha$, B, β.

La signification de ces deux règles est comme suit :

- Règle 1 exprime que lorsqu'une expression de type B / A est suivie d'une expression de type A alors nous pouvons en déduire que l'expression formée de la concaténation de ces deux expressions est de type B.
- Règle 2 exprime que lorsqu'une expression de type A \ B est précédée d'une expression de type A, alors nous pouvons en déduire que la concaténation de ces deux expressions est de type B.

Les catégories du vocabulaire, utilisé dans une grammaire catégorielle, sont des expressions (parfois complexes) utilisant les deux opérateurs de dérivation / et \. De manière non formelle, nous pouvons donner la signification suivante à ces opérateurs : Une expression de type B / A (respectivement A \ B) *attend une expression de type A à sa droite (resp. à sa gauche) pour former une expression de type B.*

La définition d'une grammaire catégorielle est comme suit :

Définition :

Une grammaire catégorielle est un quadruplet (Σ, Pr, F, s) tel que [6] :

- Σ est un ensemble fini de symboles formant le vocabulaire,
- Pr est un ensemble fini de types primitifs,
- F est une fonction de Σ vers les sous-ensembles fini de T_p, avec T_p le plus petit ensemble tel que :
  - $Pr \subseteq T_p$,
  - Si A, B $\in T_p$ alors A \ B, B \ A $\in T_p$.

  Le rôle de F est d'associer à chaque mot du vocabulaire une catégorie.
- s $\in$ Pr est la catégorie qui définit les phrases correctes du langage.

Le langage généré par une grammaire catégorielle est donné par la définition suivante :

Définition :

Le langage généré par une grammaire catégorielle (Σ, Pr, F, s) est l'ensemble :

$\{a_1, a_2, \ldots, a_n \in \Sigma^* \mid \forall\ 1 \leq i \leq n,\ \exists\ A_i : F(a_i) = A_i$ et $A_1, A_2, \ldots, A_n \rightarrow^* s\}$ avec $\rightarrow^*$ étant la fermeture réflexive et transitive de $\rightarrow$.

Exemple :

La grammaire {a : s / B, b : B, s \ B} génère le langage $\{a^n b^n \mid n > 0\}$.

Par exemple, pour n = 3, a^3b^3 peut être dérivé comme suit :

s / B, s / B, s / B, B, s \ B, s \ B →

s / B, s / B, s, s \ B, s \ B →

s / B, s / B, B, s \ B →

s / B, s, s \ B →

s / B, B →

s

3. Intérêt de l'utilisation de la grammaire catégorielle

Les grammaires catégorielles présentent deux intérêts majeurs :

1- Elles peuvent être considérées comme un système de logique. En effet, techniquement, les grammaires catégorielles, depuis leur formulation logique par Lambek en 1958 [3], et plus encore dans les extensions récentes, s'apparentent plus à la théorie de la démonstration qu'à la théorie des langages classiques, ou qu'aux autres formalismes de traitement automatique des langues ; cela explique en partie leur relatif isolement. L'autre raison de la discrétion des grammaires catégorielles jusqu'aux années 80 est que le calcul de Lambek est trop restrictif, et que nous ne connaissions pas de liens entre ce genre de calcul logique (linéaire avant l'heure) et les logiques usuelles : classique, intuitionniste et modale. Il a fallu attendre que soient établies des connexions avec des calculs logiques plus standard, c'est-à-dire que la logique s'intéresse à des calculs restreints : des liens ont premièrement été établis avec la logique modale, surtout par Van Benthem [7] et Moortgat [8], puis avec la logique linéaire inventée par Girard, qui entretient d'étroites relations avec les logiques intuitionnistes et classique [9].

2- Les grammaires catégorielles présentent un intérêt majeur pour l'apprentissage où elles permettent une lexicalisation totale. Cette dernière n'est pas qu'un plus esthétique : cette approche est d'un grand intérêt pour l'apprentissage, sujet d'importance primordiale en intelligence artificielle, dans les sciences cognitives et, sous une forme plus spécialisée, en linguistique. Les grammaires lexicalisées présentent le grand avantage de ne pas avoir de règles à apprendre, puisque celles-ci sont fixes ; il suffit alors de déterminer l'entrée lexicale associée à un nouveau mot ou au nouvel usage d'un mot déjà présent dans le lexique.

4. Apprentissage des grammaires catégorielles

4.1 Définitions préliminaires

Un mot et les différentes catégories grammaticales auxquelles il pourrait appartenir sont liés grâce à un de grammaire catégorielle appelé grammaire catégorielle k-valuée et qui est définie comme suit :

Définition :

Une grammaire catégorielle est k-valuée si à chaque mot de son vocabulaire sont associées au maximum k catégories, avec k un entier naturel fixé.

Dans la suite, nous n'allons présenter l'apprentissage que pour le cas des grammaires catégorielles *rigide*, càd, celle où *à chaque mot de son vocabulaire n'est associée qu'une seule catégorie*. Notons qu'une telle grammaire correspond à la restriction de la valeur de k à 1.

L'apprentissage se fait sur des exemples positifs, càd, des phrases bien formées, dont nous devons reconstituer les arbres de dérivation. Un arbre de dérivation pour une grammaire catégorielle (Σ, Pr, F, s) est défini de la manière suivante :

Définition :

Si D est une dérivation de B à partir de $A_1,\ldots,A_n$ et $a_1,\ldots,a_n \in \Sigma$ tels que $G : a_i \rightarrow A_i$ pour $1 \leq i \leq n$, l'arbre constitué par D aux feuilles duquel sont attachés les $a_1,\ldots,a_n$ de gauche à droite dans cet ordre, est un arbre de dérivation partiel de G.

Un arbre de dérivation complet est défini comme suit :

Définition :

Un arbre de dérivation est dit complet si la catégorie qui figure à sa racine est s.

Les noeuds d'un arbre de dérivation sont marqués par un opérateur qui désigne la règle de réduction utilisée : / et \ dans les cas suivants :

- $B / A, A \Rightarrow / B$

- $A, A \setminus B \Rightarrow \setminus B$

Exemple :

Soit la phrase suivante : *Zidane marque un but.*

Son arbre de dérivation est celui donné par Figure 1.1. Dans cette figure, N (resp. NP et S) est l'abréviation de Nom (resp. Nom Propre et Sentence).

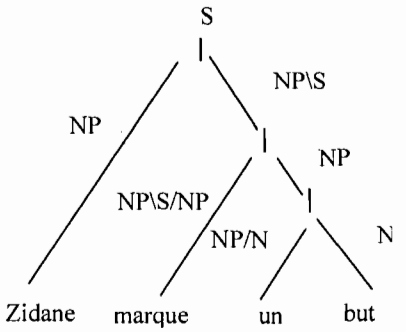


Figure 1.1 : Exemple d'un arbre de dérivation.

Le but de l'apprentissage est de reconstituer, pour des exemples donnés, leur arbre de dérivation indiquant les catégories des mots. Pour cela, il faut que la structure des arbres soit connue : nous supposons qu'elle sera fournie avec les exemples. Ceci constitue une approximation du modèle d'apprentissage.

Définition :

Une structure d'arbre de dérivation est un arbre de dérivation pour lequel les catégories des noeuds et des feuilles ne sont pas précisées.

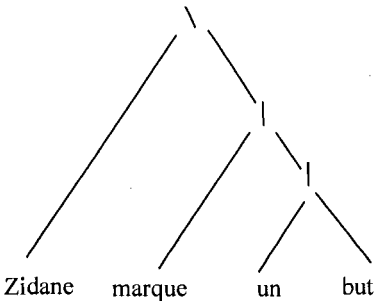
La structure d'arbre de dérivation doit donc comprendre, pour chaque noeud, l'opérateur utilisé (/ ou \).

Exemple :

Reprenons la phrase *Zidane marque un but.* Sa structure d'arbre de dérivation peut être représentée de deux manières :

[\ Zidane [/ marque [/ un but]]]

ou



4.2 L'algorithme RG

L'algorithme utilisé pour réaliser cet apprentissage est l'algorithme RG de Buszkowski et Penn [10] : il permet d'apprendre des grammaires catégorielles rigides à partir d'exemples positifs donnés sous forme de phrases avec leur structure d'arbre de dérivation.
Il comprend deux phases :

- la phase d'étiquetage,
- la phase d'unification.

4.2.1 Phase d'étiquetage

La phase d'étiquetage consiste à déduire d'une structure d'arbre de dérivation un arbre de dérivation complet, c'est-à-dire à déduire les catégories des mots en connaissant les règles utilisées. Pour cela, nous procédons en deux étapes :

1. Affecter la catégorie **s** à la racine de l'arbre puisque la phrase est correcte puis étiqueter les sous arbres grâce à l'étape 2.

2. Etiquetage d'un noeud de catégorie **A** :
   - si l'opérateur est \ (*cf.* Figure 1.2 a) :
     a. affecter au noeud de l'opérande gauche une nouvelle catégorie **B**
     b. affecter au noeud de l'opérande droit la catégorie **B \ A**
     c. étiqueter les sous arbres
   - si l'opérateur est / (*cf.* Figure 1.2 b) :
     a. affecter au noeud de l'opérande droit une nouvelle catégorie **B**
     b. affecter au noeud de l'opérande gauche la catégorie **A / B**
     c. étiqueter les sous arbres

La figure 1.2 suivante illustre les étapes précédemment décrites :

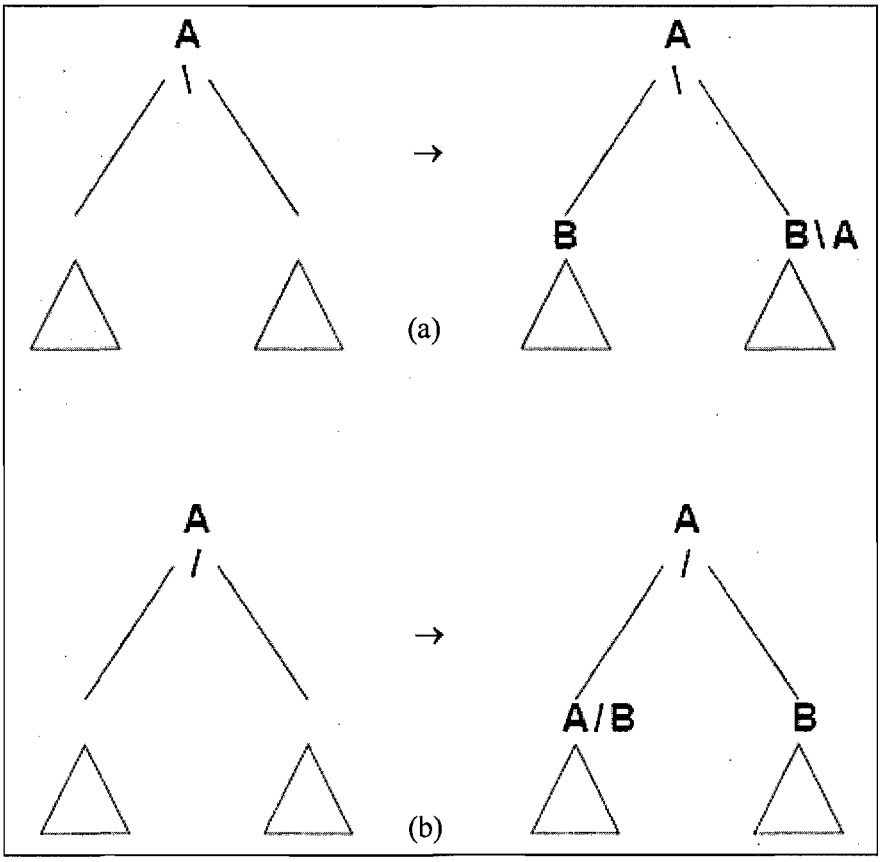


Figure 1.2 : Les étapes de la phase d'étiquetage : (a) opérateur \ et (b) opérateur /.

4.2.2 Phase d'unification

En analysant différentes phrases et grâce à la phase d'étiquetage, nous pourrions obtenir plusieurs types pour un même mot si ce dernier joue différents rôles dans ces phrases. Or les grammaires catégorielles rigides imposent un et un seul type par entrée lexicale. Ainsi, la phase d'unification, qui constitue l'élément central de l'apprentissage, identifie les différents types associés à un même mot puis substitue un type par un autre. Il est important

de noter que cette phase est inutile dans le cas ou plusieurs catégories pourraient être associées à un mot donné.

Définition :

Soient G_1, G_2 deux grammaires catégorielles. Nous disons que $G_1 \subseteq G_2$ si et seulement si pour tout $a \in \Sigma$ tel que $G_1 : a \rightarrow A$, nous avons aussi $G_2 : a \rightarrow A$.

Le processus d'unification nécessite la définition d'un opérateur de substitution noté σ et définit comme suit :

Définition :

Une substitution est une fonction σ : Var $\rightarrow$ Tp qui associe à toute variable un type. Nous pouvons étendre cette fonction à σ : Tp $\rightarrow$ Tp avec, pour tous A, B $\in$ Tp :

- σ (A/ B) = σ (A) / σ (B)
- σ (A\ B) = σ (A) \ σ (B)

Définition :

Soit G = (Σ , Pr, F, s) une grammaire catégorielle, et σ une substitution. Alors σ [G] est la grammaire obtenue par application de σ aux types de G :

- σ [G] = (Σ , Pr, F, s)
- σ [G] est une instance par substitution de G.

Définition :

Soient deux types A et B. Une substitution σ est un unifieur de A et B si σ (A) = σ (B). σ est dit unifieur le plus général de A et B si, pour tout autre unifieur τ de A et B, il existe une substitution η telle que $\tau = \sigma \circ \eta$, avec $\circ$ l'opérateur de composition. Ainsi, τ (C) = η (σ (C)), pour C = A ou C = B.

Parmi les équations entre les catégories résultant de l'identification, l'algorithme d'unification consiste à appliquer la règle correspondante parmi les règles ci-dessous :

(1) si A / B = C / D, alors A = C et B = D (idem pour \)

(2) si A / B = C \ D, alors échec

(3) si x = x, alors ne rien faire

(4) si t = x où t n'est pas un type primitif, alors remplacer par l'équation x = t

(5) si x = t et x a une autre occurrence dans l'ensemble des équations, alors :

si *x* apparaît dans t

alors échec

sinon remplacer x par t dans toutes les autres équations

Exemple :

Soit l'ensemble des exemples positifs suivants (*cf.* Figure 1.3) :

{ [\ [/ le chat] [/ regarde [/ la vache]]] , [\ [/ la vache] [/ regarde [/ le train]]] }

1. Phase d'étiquetage

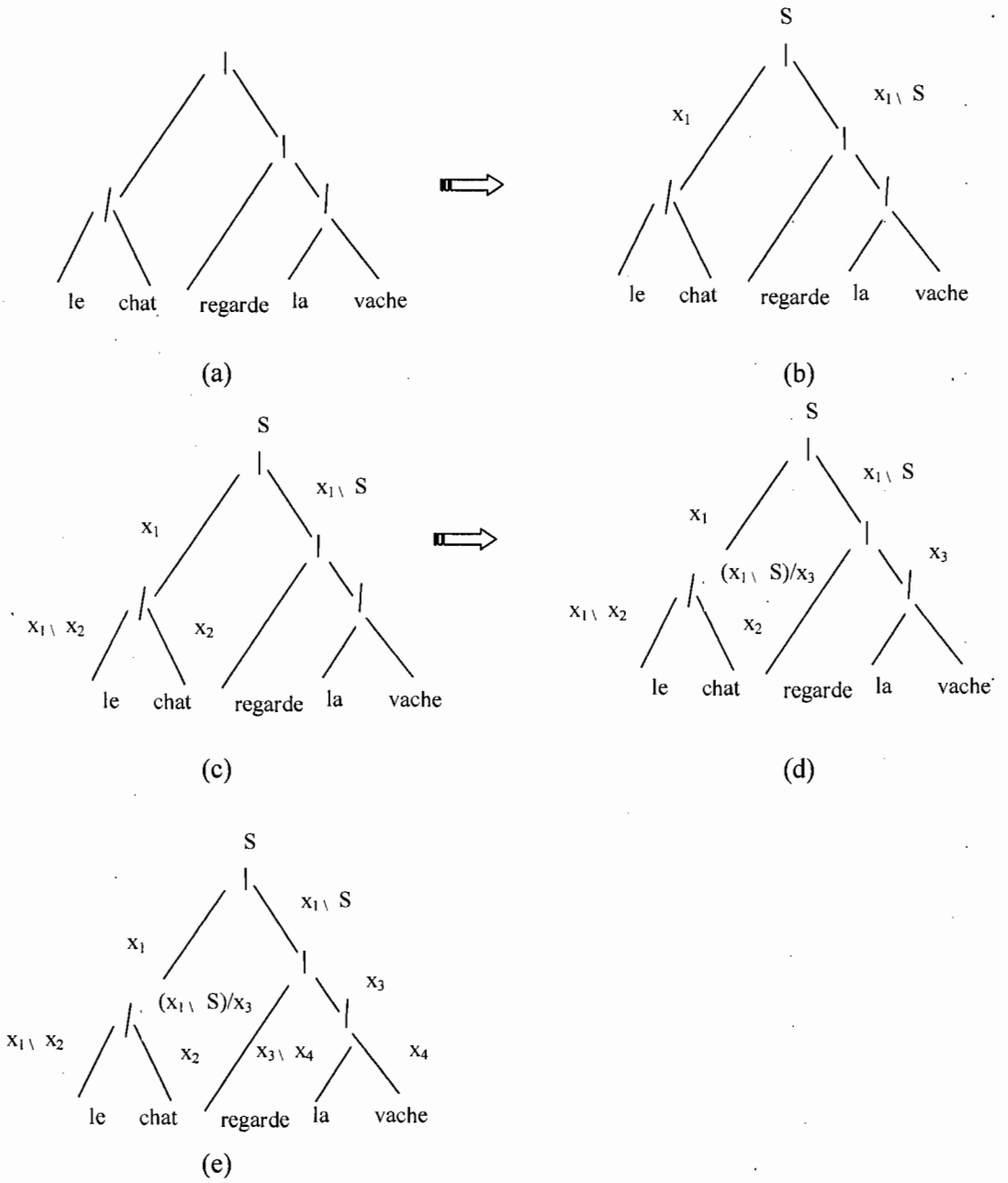


Figure 1.3 : Phase d'étiquetage de la première phrase.

Traitons par exemple l'étiquetage du nœud $x_1 \setminus S$ de la figure 1.3.(d) (et qui devient par la même occasion, racine du sous-arbre). Si nous nous référons au processus à suivre pour réaliser l'étiquetage de l'arbre, nous aurons $A = x_1 \setminus S$. Comme l'opérateur est '/', nous affectons au nœud de l'opérande droit une nouvelle catégorie B égale à x_3. L'opérande gauche quant à elle aura pour étiquette A/B c'est-à-dire $(x_1 \setminus S) / x_3$. Le processus d'étiquetage est ainsi récursif.

De la même façon, nous étiquetons la structure d'arbre de dérivation du second exemple et nous obtenons l'arbre de dérivation suivant :

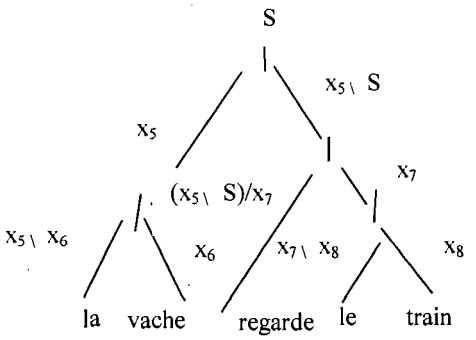


Figure 1.4 : Arbre de dérivation de la deuxième phrase.

Nous obtenons donc le lexique suivant :

- le : x_1 / x_2 , x_7 / x_8
- la : x_3 / x_4 , x_5 / x_6
- chat : x_2
- vache : x_4 , x_6
- train : x_8
- regarde : $(x_1 \setminus S) / x_3$, $(x_5 \setminus S) / x_7$

2. Phase d'unification

Nous avons plusieurs catégories pour un même mot : il faut les identifier et faire toutes les substitutions nécessaires.

- identification sur *le* : $\sigma(x_7) = x_1$ $\sigma(x_8) = x_2$

on obtient le lexique suivant :

{ ***le*** : x_1 / x_2 ; ***la*** : x_3 / x_4 , x_5 / x_6 ; ***chat*** : x_2 ; ***vache*** : x_4 , x_6 ; ***train*** : x_2 ; ***regarde*** : $(x_1 \setminus S) / x_3$, $(x_5 \setminus S) / x_1$ }

- identification sur *la* : $\sigma(x_5) = x_3$ $\sigma(x_6) = x_4$

on obtient le lexique suivant :

{ ***le*** : x_1 / x_2 ; ***la*** : x_3 / x_4 ; ***chat*** : x_2 ; ***vache*** : x_4 , x_4 ; ***train*** : x_2 ; ***regarde*** : $(x_1 \setminus S) / x_3$, $(x_3 \setminus S) / x_1$ }

- identification sur *vache* : $\sigma(x_4) = x_4$

nous obtenons le lexique suivant :

{ ***le*** : x_1 / x_2 ; ***la*** : x_3 / x_4 ; ***chat*** : x_2 ; ***vache*** : x_4 ; ***train*** : x_2 ; ***regarde*** : $(x_1 \backslash S) / x_3$, $(x_3 \backslash S) / x_1$ }

- identification sur *regarde* : $\sigma(x_3) = x_1$

nous obtenons le lexique suivant :

{ ***le*** : x_1 / x_2 ; ***la*** : x_1 / x_4 ; ***chat*** : x_2 ; ***vache*** : x_4 ; ***train*** : x_2 ; ***regarde*** : $(x_1 \backslash S) / x_1$ }

Nous obtenons finalement la grammaire (définie par son lexique) suivante :

- le : x_1 / x_2
- la : x_1 / x_4
- chat : x_2
- vache : x_4
- train : x_2
- regarde : $(x_1 \backslash S) / x_1$

Il a été démontré que **l'algorithme RG converge** : Kanazawa [11] a montré que l'unification, qui permet de ne pas ajouter un type à chaque nouvel exemple d'utilisation d'un mot, fait converger l'algorithme.

Dans la suite, nous allons présenter d'une manière succincte un type particulier de grammaire catégorielle à savoir la grammaire catégorielle combinatoire applicative.

5. Grammaire Catégorielle Combinatoire Applicative

Cette partie est principalement basée sur [12]. La Grammaire Catégorielle Combinatoire Applicative (GCCA) étend la Grammaire Catégorielle Combinatoire de Steedman [13] par une association canonique entre les règles et des combinateurs de Curry et Feys [14] d'une part et l'utilisation de métarègles qui contrôlent les opérations de changement de type d'autre part. Ce modèle est inclus dans le modèle général de la Grammaire Applicative et Cognitive [15] avec trois niveaux de représentation : (i) le phénotype (expressions concaténées) ; (ii) le génotype (expressions applicatives) ; (iii) les représentations cognitives (sens des prédicats linguistiques).

5.1 Introduction a la Grammaire Applicative et Cognitive

La Grammaire Applicative et Cognitive [15] postule trois niveaux de description des langues :

a- le niveau phénotypique (ou le phénotype) où sont représentées les caractéristiques particulières des langues naturelles (par exemple l'ordre des mots, les cas morphologiques, etc.). Les expressions linguistiques de ce niveau sont des unités linguistiques concaténées, la concaténation est notée par : 'u_1-u_2-...-u_n'.

b- le niveau génotypique (ou le génotype) où sont exprimés les invariants grammaticaux et les structures sous-jacentes aux énoncés du niveau phénotypique. Le niveau génotypique est structuré comme un langage formel appelé "langage génotype" ; il est décrit par une grammaire appelée "grammaire applicative".

c- le niveau cognitif où sont représentées les significations des prédicats lexicaux par des schèmes sémantico cognitifs.

Les trois niveaux font appel à des formalismes applicatifs typés où l'opération d'application d'un opérateur à un opérande est considérée comme primitive. Les niveaux deux et trois s'expriment dans le formalisme de la logique combinatoire typée de Curry et Feys [14]. Cette logique fait appel à des opérateurs abstraits - appelés "combinateurs" - qui permettent de composer intrinsèquement des opérateurs plus élémentaires entre eux [15]. Les combinateurs sont associés à des règles d'introduction et d'élimination. Les combinateurs que nous allons utiliser, dans cette présentation, sont **B**, C* [1], **F**, avec les règles d'élimination suivantes (U_1, U_2, U_3 et U_4 sont des expressions applicatives typées) expliqués dans la suite :

B U_1 U_2 U_3

----------(e-**B**)

U_1 (U_2 U_3)

C* U_2 U_1

----------(e- C*)

U_1 U_2

F U_1 U_2 U_3 U_4

----------(e-**F**)

[1]Le combinateur **C*** est aussi noté T.

U_1 (U_2 U_4) (U_3 U_4)

Ces règles conduisent à une autre formulation sous forme de règle de réduction (b-réduction) :

(**B** U_1 U_2) U_3) ➔ (U_1 (U_2 U_3))

((C* U_1) U_2) ➔ (U_2 U_1)

((**F** U_1 U_2 U_3) U_4) ➔ (U_1 (U_2 U_4)(U_3 U_4))

Le combinateur B permet la composition de deux expressions applicatives typées U_1 et U_2 (U_1 et U_2 fonctionnent comme opérateurs). Le résultat (B U_1 U_2) serait alors l'opérateur complexe de l'expression applicative typée U_3 (U_3 fonctionnant comme un opérande). Le combinateur C* est appliqué à une expression applicative typée U_1 (U_1 fonctionnant comme opérande de U_2). Il permet de construire l'opérateur complexe (C* U_1) de sorte à l'appliquer à l'expression applicative typée U_2. Le combinateur F permet de distribuer l'application de deux expressions applicatives typées U_2 et U_3 (fonctionnant comme des opérateurs à l'expression applicative typée U4 (fonctionnant comme opérande).

Dans la sous-section suivante, nous allons présenter la « Grammaire Catégorielle Combinatoire Applicative » (GCCA). Cette dernière est un système d'analyse formelle permettant de relier explicitement les expressions phénotypiques à leurs représentations sous-jacente dans le génotype[2]. Ces expressions représentent les deux premiers niveaux de la grammaire applicative cognitive (le phénotype et le génotype). Le système proposé par Biskri et Descles consiste en :

(i) l'analyse syntaxique des expressions concaténées du phénotype par une Grammaire Catégorielle Combinatoire.

(ii) la construction à partir du résultat de l'analyse syntaxique d'une interprétation sémantique fonctionnelle des expressions phénotypiques.

---

[2] Dans le phénotype, les expressions linguistiques sont concaténées selon les règles syntagmatiques du français. Dans le génotype, les expressions sont agencées selon l'ordre applicatif.

5.2 La Grammaire Catégorielle Combinatoire et Applicative

Les Grammaires Catégorielles assignent des catégories syntaxiques à chaque unité linguistique. Les catégories syntaxiques sont des types orientés engendrés à partir de types de base et des deux opérateurs constructifs '/' et '\'.

(i) N (syntagme nominal) et S (phrase) sont des types de base.

(ii) Si X et Y sont des types orientés alors X / Y et X \ Y sont des types orientés [3]. Une unité linguistique u de type orienté X sera désigné par '[X : u]'.

Les deux règles d'application (avant et arrière) sont notées:

$[X / Y : u_1] - [Y : u_2]$		$[Y : u_1] - [X \setminus Y : u_2]$
---------->	;	<----------
$[X : (u_1\ u_2)]$		$[X : (u_2\ u_1)]$

Les prémisses dans chaque règle sont des concaténations d'unités linguistiques à types orientés considérées comme étant des opérateurs ou des opérandes, la conséquence de chaque règle est une expression applicative avec un type orienté.

La Grammaire Catégorielle Combinatoire [12] généralise les Grammaires Catégorielles classiques en introduisant des opérations de changement de type et des opérations de composition des types fonctionnels. Les nouvelles règles proposées visent une analyse pseudo incrémentale (de gauche à droite) pour éliminer le problème de la pseudo-ambiguïté [16, 17]. En effet, une phrase non ambiguë peut avoir plusieurs analyses syntaxiques possibles qui ne correspondent qu'à une seule interprétation sémantique [5], chose qui n'est pas le cas pour une phrase ambiguë. Pour ce dernier cas, le rôle des règles susmentionnées est de n'autoriser qu'une seule stratégie d'analyse syntaxique qui consiste à parcourir les textes de gauche à droite.

Nous considérons dans la GCCA que les règles de la Grammaire Catégorielle Combinatoire de Steedman introduisent les combinateurs **B**, **C*** dans la séquence syntagmatique. Cette introduction permet de passer d'une structure concaténée à une structure applicative.

---

[3] Nous choisissons ici la notation de Steedman [13] : X / Y et X \ Y sont des types orientés fonctionnels. Une unité linguistique 'u' avec le type X / Y (respectivement X \ Y) est considérée comme un opérateur (ou une fonction) dont l'opérande de type Y est positionné à droite (respectivement à gauche) de l'opérateur.

Les règles de la GCCA sont :

1- Règles de changement de type :

[X : u]		[X : u]
----------->**T**	;	-----------<**T**
[Y/(Y\X) : (C* u)]		[Y\(Y/X) : (C* u)]
[X : u]		[X : u]
----------->**Tx**	;	-----------<**Tx**
[Y/(Y/X) : (C* u)]		[Y\(Y\X) : (C* u)]

2- Règles de composition fonctionnelle :

[X/Y : u_1]-[Y/Z : u_2]		[Y\Z : u_1]-[X\Y : u_2]
----------->**B**	;	-----------<**B**
[X/Z : (**B** u_1 u_2)]		[X\Z : (**B** u_2 u_1)]
[X/Y : u_1]-[Y\Z : u_2]		[Y/Z : u_1]-[X\Y : u_2]
----------->**Bx**	;	-----------<**Bx**
[X\Z : (**B** u_1 u_2)]		[X/Z : (**B** u_2 u_1)]

Les prémisses des règles sont des expressions concaténées typées ; les résultats sont des expressions applicatives (typées) avec éventuellement introduction d'un combinateur. Le changement de type d'une unité u introduit le combinateur **C***; la composition de deux unités concaténées introduit le combinateur **B**. Avec ces règles nous pouvons analyser une phrase au moyen d'une stratégie quasi-incrémentale "de gauche à droite". Le choix d'une telle stratégie est motivé par le contrôle du problème de la pseudo-ambiguïté [13,17].

Exemple :

Jean	-	*aime*
-------		-------
[N : *Jean*]		
----------->**T**		
[S/(S\N) : (C* *Jean*)]	-	[(S\N)/N : *aime*]
----------->**B**		

[S/N : (**B** (C* *Jean*) *aime*)]

La première règle (**>T**) appliquée à l'unité typée [N : *Jean*] transforme l'opérande en opérateur. Elle construit une structure applicative (C* *Jean*) ayant pour type S/(S\N). L'introduction du combinateur C* illustre dans la représentation applicative le changement de type : (C* *Jean*) fonctionne comme un opérateur avec son type fonctionnel. La règle (**>B**) combinent les unités linguistiques typées [S/(S\N) : (C* *Jean*)] et [(S\N)/N : *aime*] avec le combinateur **B** de façon à pouvoir composer les deux unités fonctionnelles (C* *Jean*) et *aime*.

Un traitement complet basé sur la Grammaire Catégorielle Combinatoire Applicative s'effectue en deux grandes étapes :

(i) la première étape s'illustre par la vérification de la bonne connexion syntaxique et la construction de structures prédicatives avec des combinateurs introduits à certaines positions de la chaîne syntagmatique,

(ii) la deuxième étape consiste à utiliser les règles de b-réduction des combinateurs de façon à former une structure prédicative sous-jacente à l'expression phénotypique. L'expression obtenue est applicative et appartient au langage génotype. La GCCA engendre des processus qui associent une structure applicative à une expression concaténée du phénotype. Il nous reste à éliminer les combinateurs de l'expression obtenue de façon à construire la "forme normale" (au sens technique de la b-réduction) qui exprime l'interprétation sémantique fonctionnelle. Ce calcul s'effectue entièrement dans le génotype.

Le traitement que nous proposons donc prend la forme d'une compilation dont les étapes sont résumées dans la figure 1.5.

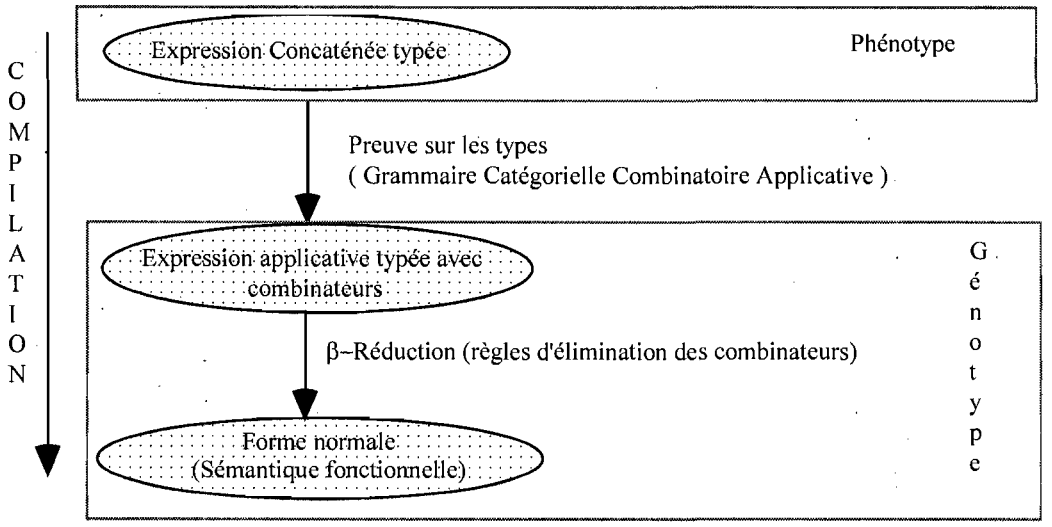


Figure 1.5 : Traitement basé sur la Grammaire Catégorielle Combinatoire Applicative.

Exemple :

Soit la phrase suivante : *Jean aime Marie*

1	[N:*Jean*]-[(S\N)/N:*aime*]-[N:*Marie*]		**Structure concaténée typée du phénotype**
2	[S/(S\N):(**C*** *Jean*)]-[(S\N)/N:*aime*]-[N:*Marie*]		(**>T**)
3	[S/N:(**B** (**C*** *Jean*) *aime*)]-[N:*Marie*]	(**>B**)	**Compilation**
4	[S:((**B** (**C*** *Jean*) *aime*) *Marie*)]	(>)	
	V		
5	[S : ((**B** (**C*** *Jean*) *aime*) *Marie*)]		**Structure applicative typée du génotype**
6	[S : ((**C*** *Jean*) (*aime Marie*))]	(**B**)	
7	[S : ((*aime Marie*) *Jean*)]	(**C***)	
	Forme normale du génotype		

Le changement de type (**>T**) affectant l'opérande *Jean* permet d'engendrer l'opérateur (**C*** *Jean*) que la règle fonctionnelle (**>B**) compose avec l'opérateur *aime*. L'opérateur complexe (**B** (**C*** *Jean*) *aime*) s'applique à l'opérande *Marie* pour former l'expression applicative du génotype ((**B** (**C*** *Jean*) *aime*) *Marie*). La réduction des combinateurs dans le génotype construit l'interprétation sémantique fonctionnelle sous-jacente à l'expression phénotypique en entrée.

6. Conclusion

Dans ce chapitre, nous avons présenté le cadre formel des grammaires catégorielles ainsi que l'intérêt pratique de leur utilisation. Ensuite, nous avons détaillé le cas de l'apprentissage dans le cadre des grammaires catégorielles rigides. Ces dernières sont considérées comme un cas particulier des grammaires catégorielles k-valuées lorsque k = 1. Enfin, nous avons présenté la grammaire catégorielle combinatoire applicative. Cette dernière étend la grammaire cognitive applicative.

Chapitre 2 : Théorie des graphes

1. Introduction

Un graphe se définit simplement comme un ensemble de points dont certains sont reliés par des lignes. Un des exemples les plus répandus est le problème dit « du voyageur de commerce » : il s'agit de tracer le plus court chemin, càd le minimum de distance, que pourrait emprunter un représentant pour rendre visite à ses clients dans une série de villes, en ne passant qu'une seule fois dans chaque ville. Cette méthode mathématique est une branche de la combinatoire ; développée par des théoriciens à la fin du XIXe siècle, elle a trouvé des applications dans le calcul des probabilités avant d'être profondément renouvelée dans les années 60 (notamment par Claude Berge en France). Ses applications actuelles sont orientées vers l'informatique, d'où son intérêt grandissant [18].

Les graphes ont récemment fait leur entrée dans les programmes de mathématiques de l'enseignement secondaire et dans de nombreux cursus post-bac.

La théorie des graphes [19, 20] est régulièrement évoquée pour résoudre des problèmes classiques (la promenade sur les ponts de Königsberg, la coloration de cartes géographiques) ou d'autres problèmes liés au fonctionnement de notre société (transport, réseaux de communication, architectures informatiques). Si elle convainc par son utilité pratique, nous pouvons légitimement nous demander en quoi des objets aussi pauvres - des points reliés par des lignes - peuvent engendrer des problématiques incontestablement riches.

2. Généralités sur les graphes

Les graphes sont utilisés comme modèles de situations très variées. Ils permettent de représenter simplement différents problèmes ayant une écriture formelle compliquée [21].

2.1 Graphes orientés

Un graphe orienté G est donné par:

* Un ensemble X fini donnant les sommets de G.
* Un ensemble $U \subset X*X$ formé par des couples ordonnés.

U donne les arcs du graphe G. Le graphe est alors note G = (X,U).

Soit u = (i, j)∈ U, i (resp. j) représente l'extremité initiale (resp. finale) de u. i est un précèdent de j et j est un suivant de i.

Graphiquement, les sommets sont représentés par des points et le arc par des flèches reliant les sommets. Un exemple est donné par la figure 2.1.

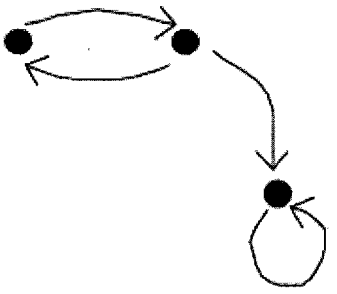


Figure 2.1 : Exemple de graphe orienté.

2.2 Graphes non orientés

Dans certains cas, la distinction entre extrémités initiale et terminale d'un arc ne joue aucun rôle. Nous nous intéresserons à l'existence ou non d'une liaison U alors appelé l'ensemble des arêtes.

2.3 Degré

Dans un graphe non orienté, le degré d'un sommet est le nombre d'arêtes auxquelles ce sommet appartient. La somme des degrés de chaque sommet est égale au double du nombre total d'arêtes.

Dans un graphe orienté, nous distinguons pour un sommet *s* le degré entrant et le degré sortant. Le premier correspond au nombre d'arcs dont l'extrémité finale est *s*. Le second est le nombre d'arcs dont l'extrémité initiale est *s*. Le degré d'un sommet *s* dans un graphe orienté est la somme du degré entrant et sortant de *s*.

Le degré maximum (resp. minimum) d'un graphe *G*, noté $\Delta(G)$ (resp. $\delta(G)$) est égal au degré maximum (resp. minimum) dans l'ensemble des degrés de tous les sommets de *G*.

2.4 Classes de graphes

2.4.1 Arbres

Un arbre est un graphe connexe non orienté et acyclique (sa forme évoque en effet la ramification des branches d'un arbre). Deux types de sommets dans un arbre sont à distinguer :

- les feuilles,
- les nœuds internes.

La figure 2.2 donne un exemple de ces deux types de sommets. En effet, les nœuds 1, 3, 5 et 7 représentent les 4 feuilles de l'arbre, tandis que le reste des nœuds (càd 2, 4 et 6) sont des nœuds internes.

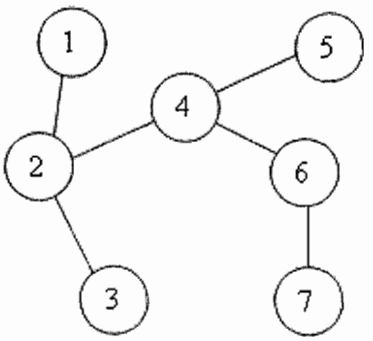


Figure 2.2 : Un arbre avec 4 feuilles (nœuds 1, 3, 5 et 7) et 3 nœuds internes (nœuds 2, 4 et 6).

2.4.2 Arborescences

Une arborescence est un graphe orienté acyclique connexe où chaque sommet est de degré entrant au plus 1. Dans ce cas un seul sommet est de degré entrant 0, il est appelé **racine** de l'arborescence.

2.4.3 Chaînes

Les chaînes sont des arbres possédant 2 feuilles.

2.4.4 Chemins

Graphe orienté connexe de degré entrant maximum 1, de degré sortant maximum 1 et de degré minimum 1.

2.4.5 Cycles et circuits

Un graphe G non orienté (resp. orienté) et connexe est un cycle (resp. circuit) si et seulement si tous les sommets sont de degré 2 (resp. de degré entrant 1 et de degré sortant 1).

2.4.6 Forêt

Une forêt est un graphe acyclique. Chacune de ses composantes connexes est donc un arbre.

2.4.7 Graphe aléatoire

Un graphe est aléatoire s'il est généré par un processus aléatoire.

2.4.8 Graphe biparti

Un graphe est biparti s'il y a une partition des sommets du graphe en deux sous ensembles A et B telle que toutes les arêtes du graphe ont un sommet dans A et un sommet dans B.

2.4.9 Graphe complet

Un graphe complet est un graphe dont tous les sommets sont reliés deux à deux. Le graphe complet à n sommets est noté: K_n.

2.4.10 Graphe connexe

Un graphe non orienté est connexe, si et seulement si pour toute paire de sommets [a, b] il existe une chaîne entre les sommets a et b.
Si nous parlons de connexité pour un graphe orienté, c'est que nous considérons non pas ce graphe, mais le graphe non orienté correspondant.

2.4.11 Graphe k-connexe

Un graphe non orienté est k-connexe s'il reste connexe après suppression d'un ensemble quelconque de k-1 arêtes et s'il existe un ensemble de k arêtes qui déconnecte le graphe. Autrement dit, un graphe est k-connexe si et seulement s'il existe au moins k chaînes indépendantes (arcs-disjointes) entre chaque couple de sommets. Cette notion est utilisée en

électronique, en calcul de la fiabilité, et dans l'étude de jeux de stratégie comme le *cut and connect*.

2.4.12 Graphe fortement connexe

Un graphe orienté est dit fortement connexe, si pour tout couple de sommets (u, v) du graphe il existe un chemin de *u* à *v* et de *v* à *u*. Les travaux de McCullogh et Pitts [22] suggèrent que le cerveau des mammifères est fortement connexe.

2.4.13 Graphe hamiltonien

Graphe qui contient un cycle hamiltonien. Le graphe est nommé *hypohamiltonien* s'il suffit d'en retirer un sommet quelconque pour qu'il devienne hamiltonien. Utile dans le problème du voyageur de commerce.

2.4.14 Graphe d'intervalles

Si I est un ensemble d'intervalles sur les réels et que nous pouvons associer à chaque sommet un intervalle et que pour chaque sommet u et v il y a une arête entre u et v si et seulement si l'intersection entre leurs intervalles associés n'est pas nulle, alors le graphe défini par ces sommets et ces arêtes est un graphe d'intervalles.

2.4.15 Graphe de permutation

Soit P une permutation de la séquence 1,..., n. Le graphe d'inversion associé à P est le graphe dont les sommets sont les entiers 1,..., n et dont pour tout sommets u et v il y a une arête entre u et v si et seulement si u et v sont inversés dans P.
Un graphe est un graphe de permutation s'il existe une permutation dont le graphe est son graphe d'inversion.

2.4.16 Graphe planaire

Un graphe est planaire s'il existe *au moins une* façon de le dessiner dans le plan sans que deux arêtes se croisent. Cette propriété est importante pour les circuits imprimés monocouche.

2.4.17 Graphe k-régulier

Un graphe régulier est un graphe où chaque sommet est de degré k.

2.4.18 Graphe split

Un graphe est split s'il existe une partition des sommets du graphe en deux sous-ensembles S et C telle que S est un ensemble stable et C est une clique.

2.4.19 Graphe triangulé

Un graphe est triangulé s'il ne contient pas un cycle de longueur quatre sans corde comme mineur. Les arbres, et les graphes d'intervalles, notamment, sont triangulés.

3. Chemins de longueurs extrémales

De toutes les opérations que nous effectuons sur un graphe pondéré (ou valué), une retient l'attention : trouver le chemin le moins coûteux entre deux noeuds d'un graphe. Par exemple, si un graphe modélise un réseau routier : les sommets sont des villes et les arcs sont des routes liant deux villes. Le poids accordé à chaque arc est le kilométrage effectué en empruntant cette route. Quel sera l'algorithme pour trouver le chemin le plus optimal en kilométrage pour se rendre d'une ville à une autre?

3.1 Notions fondamentales

Définition :

Étant donné un graphe $G = (X, U)$, nous associons à chaque arc du graphe $u \in U$ un nombre $l(u) \in R$ appelé longueur de l'arc u. Nous disons que G est valué par les longueurs $l(u)$. Nous définissons la longueur d'un chemin μ joignant le sommet i au sommet j par : $l(\mu) = \Sigma\, l(u)$ pour tout $u \in \mu$ [23].

Dans le cas de graphe non valué (*cf.* Figure 2.3), la longueur d'un chemin est le nombre d'arcs composant ce chemin.

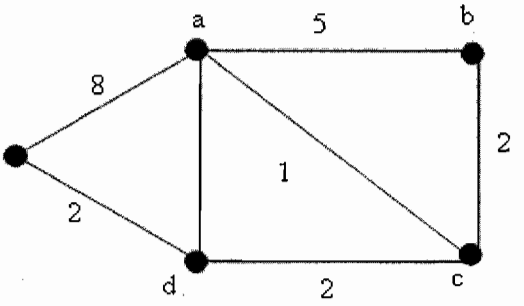


Figure 2.3: Exemple d'un graphe valué.

Définition :

Le descendant d'un sommet k est tout sommet i ∈ X tel qu'il existe un chemin de k vers i. Nous notons D(k)={descendants de k}.

Définition :

Un chemin reliant le sommet k au sommet i, de longueur extrémale, est dit ki-extrémal. Les arcs de G qui sont empruntés par un chemin extrémal d'extrémité initiale k sont appelés des arcs k-extrémaux.

Théorème :

Soit μ un chemin ki-extrémal. Toute partie μ_1 de μ reliant un sommet h à un sommet l est alors hl-extrémale

3.2 Les algorithmes

3.2. 1 Algorithme de Dijkstra dans le cas où les poids sont positifs

Soit un graphe G(V, E) pondéré avec des poids non négatifs, pour obtenir les chemins les moins coûteux à partir d'un sommet, l'algorithme de Dijkstra [23,24] répondra à cette requête. Les variables utilisées par l'algorithme présenté plus loin sont :

- *V* : l'ensemble des sommets du graphe;
- *E* : l'ensemble des arcs (ou arêtes) du graphe;
- *T* : un ensemble temporaire de sommets d'un graphe (pour ne pas modifier *V*);
- *S* : l'ensemble des sommets traités par l'algorithme (pour ne pas traiter 2 fois le même sommet);

- y : un tableau de coût de longueur $|V|$ (le nombre de sommets du graphe);
- a : l'indice du noeud de départ;
- $c(u, v)$: permet d'obtenir le coût d'un arc du sommet u au sommet v appartenant à E.

Algorithme 2.1 : Algorithme de Dijkstra

POUR sommets i de V **FAIRE**
début
    $yi \leftarrow +\infty$
fin
$ya \leftarrow 0$
$S \leftarrow \varnothing$
$T \leftarrow V$
TANT QUE $T \neq \varnothing$ **FAIRE**
début
    Sélectionner le sommet j de T de plus petite valeur yj
    $T \leftarrow T \setminus j$
    $S \leftarrow S \cup j$
    POUR sommets k de T adjacents à j **FAIRE**
    Début
        $yk \leftarrow \min(yk, yj + c(j,k))$ {$c(j,k)$ est le coût de j à k}
    fin
fin

L'idée de base est le principe de sous optimalité. Ce principe s'énonce comme suit : pour tout sommet x qui est sur le chemin le moins coûteux reliant le sommet u au sommet v, le sous-chemin de u à x est le chemin le moins coûteux reliant le sommet u au sommet x.

Prenons l'exemple du graphe non orienté décrit à la figure 2.4 :

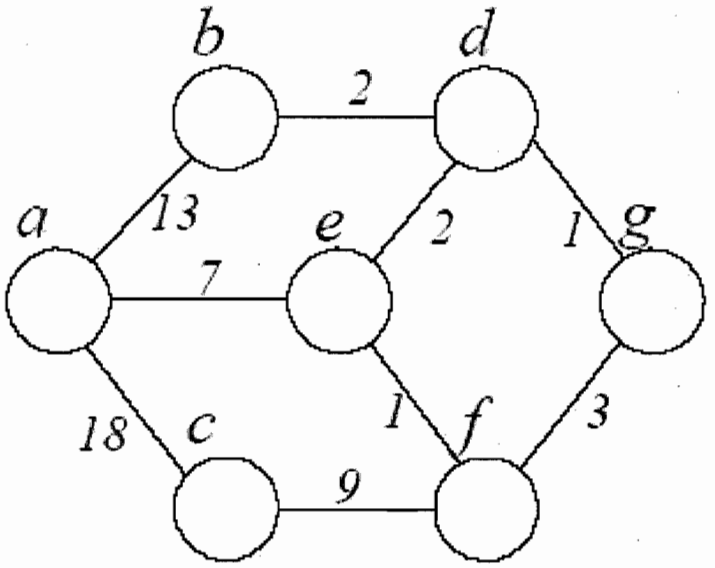


Figure 2.4 : Exemple du graphe non orienté.

Nous désirons savoir quels sont les chemins les plus courts (en fonction du coût) pour se rendre, à partir du sommet *a*, aux autres sommets du graphe. La première étape est d'initialiser le coût à la plus grande valeur possible pour tous les sommets (ce qui est représenté ici par le symbole de l'infini $+\infty$). Comme les chemins recherchés débutent au sommet *a*, le chemin le moins coûteux pour se rendre à *a* est de ne pas emprunter de chemin. La valeur de ce chemin est donc nulle. La situation initiale est modélisée par la partie *(1)* de la figure 2.5. L'ensemble *S* des sommets fixés est représenté par les sommets noirs. À l'état initial, cet ensemble est vide, donc il n'y a aucun sommet noirci. L'ensemble *T* des sommets non choisis est représenté par les sommets non noircis. Donc lorsqu'un sommet est choisi parce qu'il a le coût le plus faible, nous le transférons de l'ensemble *T* à l'ensemble *S* ou graphiquement nous le noircissons. Le coût du chemin qui débute à *a* et qui se termine à ce sommet (les *y* de l'algorithme) est le nombre dans les cercles.

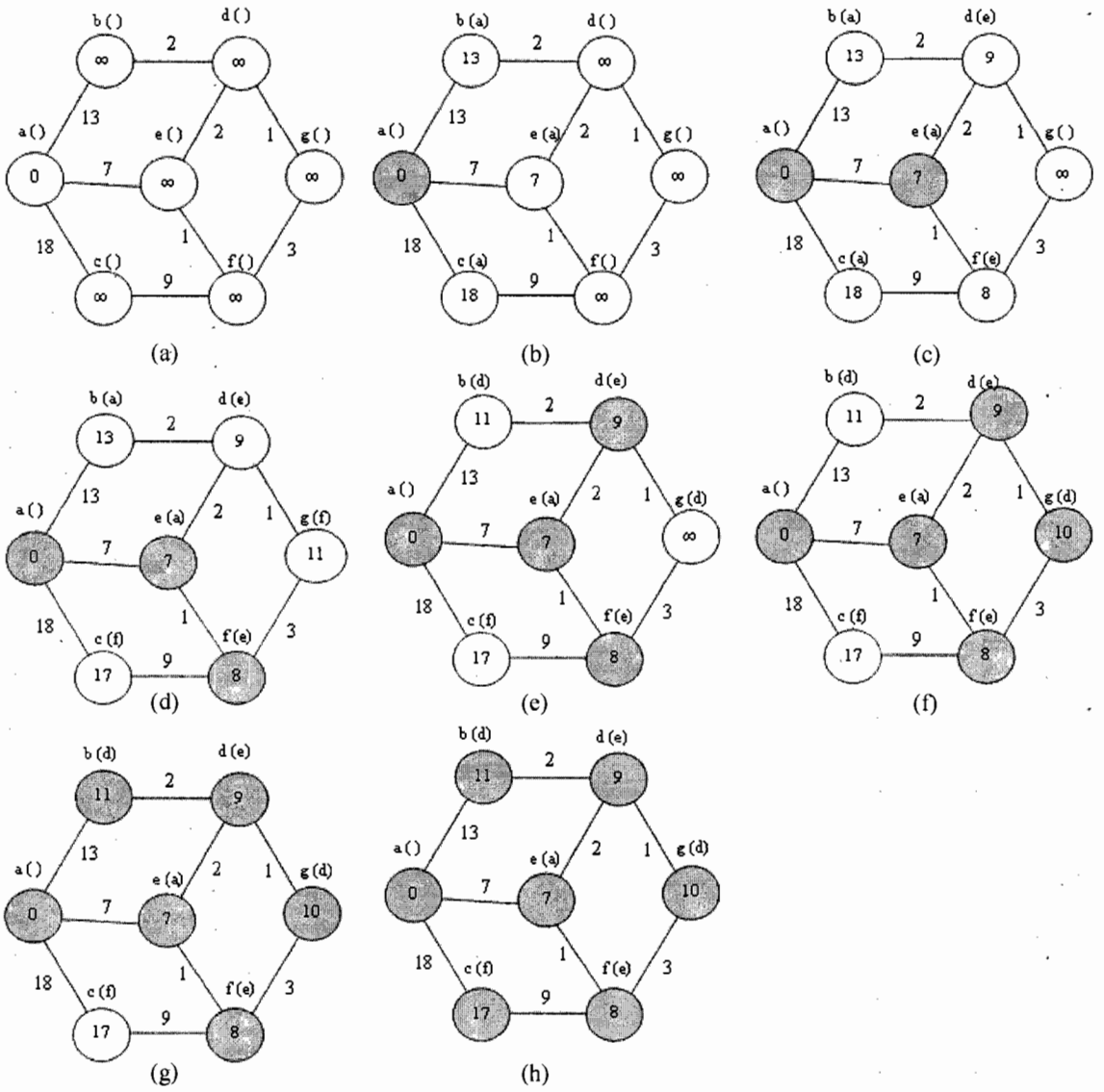


Figure 2.5 : Application de l'algorithme Dijkstra.

Il faut déterminer le chemin le plus court trouvé jusqu'à maintenant, donc il faut fixer le sommet non noirci dont le coût inscrit à l'intérieur du cercle est minimal. À ce stade, le sommet qui répond à cette définition est le sommet *a*.

Il faut alors faire la mise à jour du coût pour se rendre aux sommets adjacents. Comme il existe des arêtes entre *a* et les sommets *b*, *c* et *e*, effectuons la mise à jour des coûts pour ces sommets. Pour le sommet *b*, le coût actuel est infini et le nouveau (en passant par *a*) est de 13. Nous choisissons le minimum entre les deux, donc 13, une nette amélioration. De même, *c* et *e* qui ont tous les deux un coût actuel infini passeront respectivement à 18 et 7. Comme l'intérêt n'est pas seulement porté au coût, mais également au chemin, il doit y avoir une mémorisation du sommet qui a permis l'obtention d'un tel coût. Dans la figure 2.5, cette

origine est représentée par l'étiquette entre les parenthèses à droite de l'étiquette du sommet. Donc pour les trois sommets modifiés à cette étape, leur origine est *a*.

De l'ensemble *T*, nous choisissons le sommet ayant le coût le plus faible, c'est-à-dire le sommet *e*. Comme les sommets *d* et *f* sont les deux adjacents de *e*, nous effectuons les mises à jour suivantes: $yd = \min(yd, ye + 2) = \min(+\infty, 7 + 2) = 9$ et $yf = \min(yf, ye + 1) = \min(+\infty, 7 + 1) = 8$.

Une fois l'exécution de l'algorithme de Dijkstra terminée, nous pouvons extraire les chemins de moindre coût débutant à *a* et menant à tout autre sommet du graphe.

Une autre façon de voir le problème est d'utiliser une représentation vectorielle. La table 2.1 montre étape par étape (ligne par ligne) l'évolution de notre graphe. Une colonne est l'évolution des données sur un sommet à travers le temps. Les informations pertinentes sont l'étiquette, le coût et l'origine.

	Sommet	a	b	c	d	e	f	g
(1)	Origine:	nil	nil	nil	nil	nil	nil	nil
	Coût:	0	∞	∞	∞	∞	∞	∞
(2)	Origine:		a	a	nil	a	nil	nil
	Coût:		13	18	∞	7	∞	∞
(3)	Origine:		a	a	e		e	nil
	Coût:		13	18	9		8	∞
(4)	Origine:		a	f	e			f
	Coût:		13	17	9			11
(5)	Origine:		d	f				d
	Coût:		11	17				10
(6)	Origine:		d	f				
	Coût:		11	17				
(7)	Origine:			f				
	Coût:			17				
(8)	Origine:	nil	d	f	e	e	e	d
	Coût:	0	11	17	9	7	8	10

Table 2.1 : Exécution de l'algorithme Dijkstra.

Tout comme le parcours en largeur, nous pouvons retrouver le chemin en inversant l'ordre du chemin obtenu en partant du sommet terminal désiré et en remontant les origines jusqu'au sommet *a*. Par exemple, le chemin le moins coûteux entre le sommet *a* et le sommet *b* vaut 11. En remontant les origines à partir de *b*, le chemin, est *b* → *d* → *e* → *a*. Donc dans l'ordre, le chemin le plus court en fonction des poids est *a-e*, *e-d*, *d-b*.

3.2.2 Algorithme de Bellman-Ford dans le cas où les poids sont négatifs

Le second algorithme qui répond à la question du plus court chemin dans des temps plus raisonnables est celui de Bellman-Ford [19]. L'apport de cet algorithme à celui de Dijkstra est la possibilité d'avoir des poids négatifs entre les sommets. De tels poids – négatifs – peuvent permettre, par exemple, de tenir compte du sens emprunté sur une route en associant un poids positif à un sens donné et un poids négatif au sens contraire. Les variables utilisées dans l'algorithme de Bellman-Ford [23,24] sont :

- V : l'ensemble des sommets du graphe;
- E : l'ensemble des arcs (ou arêtes) du graphe;
- y : un tableau de coût de longueur $|V|$ (le nombre de sommet du graphe);
- a : l'indice du noeud de départ;
- c(u, v) : permet d'obtenir le coût d'un arc du sommet u au sommet v appartenant à E.

Algorithme 2.2 : Algorithme de Bellman-Ford

POUR sommets i de V **FAIRE**
début
    $yi \leftarrow +\infty$
    $ya \leftarrow 0$
    RÉPÉTER $|V| - 1$ **FOIS**
    Début
        POUR (u, v) de E **FAIRE**
        début
        $yv \leftarrow \min(yv, yu + c(u, v))$
        fin
    fin
    POUR (u, v) de E **FAIRE**
    Début
        SI $yv > yu + c(u, v)$ **ALORS FAIRE**
        Début
        RETOURNER FAUX
        fin
    fin
RETOURNER VRAI

Tout comme l'algorithme de Dijkstra, la première étape est d'initialiser les coûts à l'infini et de mettre celui du sommet de départ à 0. Par la suite, l'algorithme de Bellman-Ford fait une mise à jour des coûts pour chaque sommet (donc vérifie pour chaque arc) et ce, autant de fois que le nombre de sommets du graphe moins un. Ce qui a pour effet de stabiliser le coût de tous les chemins à moins qu'il y ait un cycle de poids négatif.

En entrant dans un cycle négatif - représentant par exemple un ensemble de routes déjà empruntées - à chaque tour de boucle, le coût diminue. Dans une telle situation, il est possible d'atteindre un coût négativement infini, alors il n'existe pas de chemin au coût minimal et l'algorithme retourne FAUX. Sinon, cela signifie que les coûts des chemins ont convergé et que les chemins obtenus sont les plus courts donc il retourne VRAI.

Cet algorithme est une amélioration de Dijkstra puisqu'il accepte les poids négatifs. Par contre, il demande un temps d'exécution plus grand.

4. Conclusion

L'objet de ce chapitre est la présentation succincte de la théorie fondamentale de la théorie de graphe et notamment les différents types de graphes. Il a souligné en particulier l'intérêt des algorithmes de recherche du chemin le plus court : Dijkstra et Bellman-Ford, qui ont été illustrés par des exemples.

Chapitre 3 : Chaînes de Markov

1. Introduction

La modélisation stochastique permet l'utilisation des modèles probabilistes pour traiter les problèmes à information incertaine ou incomplète. Ainsi, les modèles de Markov connaissent un regain d'intérêt tant dans leurs aspects théoriques qu'appliqués. La théorie des chaînes de Markov est née en 1913, une première application a été développée par Markov pour analyser le langage [27]. Ces travaux ont été utilisés régulièrement mais les premières applications exploitables furent réalisées dès les années soixante, telles que les modèles probabilistes d'urnes par Cave et Neuwirch [28], le calcul direct du maximum de vraisemblance ou l'observation de la suite d'états dans une chaîne de Markov. Ceci a permis à la communauté scientifique d'exploiter pleinement le potentiel de ces modèles. C'est dans les années 70 que des chercheurs ont apporté des algorithmes puissants permettant de résoudre les problèmes de reconnaissance, d'analyse et d'apprentissage.

2. Théorie des Chaînes de Markov

Un processus stochastique est un phénomène où intervient le hasard. Nous définissons alors $X(t)$ comme étant une variable aléatoire évoluant en fonction du temps. Exemple : une suite de lancers de dés 1, 6, 2, 5 d'où 1, 6, 2, 5 $X_0 = X_1 = X_2 = X_3$. Ce processus est dit markovien si son évolution ne dépend pas de son passé, mais uniquement de son état présent (Ceci est appelé la propriété de Markov). Un processus markovien peut être modélisé par un modèle théorique dit « Modèle de Markov ». Il existe 2 types de modèles : observable et caché [25].

2. 1 Chaîne de Markov observable

L'évolution du processus de Markov peut être représentée par un graphe de transitions d'états qui fait apparaître la structure du processus selon les règles suivantes [26, 27] :

- Les états sont représentés par des sommets (État n). Nous parlons d'alphabet des états: S = $\{s_1, s_2, s_3\}$ les états de la chaîne de Markov.

- Les transitions (possibilité de passer d'un état à un autre) sont représentées par des arêtes. Elles sont pondérées par leur probabilité (Flèche). Les probabilités sont regroupées dans

une matrice de transition: $A = \{a_{ij} = P(s_j/s_i)\}$; $\Sigma\, a_{ij} = 1$.

- Les probabilités de départ: ce sont les probabilités de débuter dans un état ou un autre (point 0). Elles sont regroupées dans un vecteur d'initialisation :

$$\Pi = \{\pi_i = P(s_i)\} \sum_{i=1}^{N} \pi_i = 1. \tag{3.1}$$

Un modèle λ est dit observable car les états sont directement observables. Il est caractérisé par une matrice de transition A et un vecteur d'initialisation Π, nous notons:

$$\lambda = \{\Pi, A\} \tag{3.2}$$

2.2 Chaîne de Markov caché

Dans un Modèle de Markov caché les états $S = \{s_1, s_2, ..., s_m\}$ sont non observables. Cependant, ils émettent des signaux observables $O = (o_1, o_2, ..., o_k)$ qui sont pondérés par leur probabilité. Le modèle λ peut être représenté graphiquement, avec [27] :

- La matrice de transitions : $A = \{a_{ij} = P(s_j/s_i)\}$; $\Sigma\, a_{ij} = 1$.
- Le vecteur d'initialisation : $\Pi = \{\pi_i = P(s_i)\} \sum_{i=1}^{N} \pi_i = 1.$
- Les probabilités que l'état s_i émettent le signal d'observation O_k (Flèche brisée). Elles sont regroupées dans une matrice d'émission :

$$B = \{b_i(o_k) = P(o_k/s_i)\}; \sum_{j=1}^{T} b_i(o_j) = 1. \tag{3.3}$$

Les représentations mathématiques seront utilisées tout au long de ce chapitre. Nous utiliserons ainsi:

- N : le nombre d'états $N = \Sigma\, S_i$
- T : le nombre d'observations possibles $T = \Sigma\, O$. Si T est défini et dénombrable, nous parlons aussi d'alphabet;
- q : l'état du système au temps t;
- M : taille de la séquence observée.

Un modèle de Markov caché λ est caractérisé par une matrice de transition A, une matrice d'observation B et un vecteur d'initialisation Π, nous notons:

$$\lambda = \{\Pi, A, B\} \tag{3.4}$$

Il est important de noter qu'un Modèle de Markov Observable peut être modélisé sous la forme d'un modèle de Markov caché où les états correspondent aux événements observés. C'est-à-dire que chaque état s_i à une probabilité '1' d'émettre l'observation b_i.

Afin de mieux appréhender les modèles de **Markov** Observables, un exemple intéressant est développé dans [25]. Le système modélisé est relatif à la météo du jour. Ceci nous permet notamment d'émettre une probabilité sur :

- Probabilité de réalisation d'une séquence.
- Prévision d'un état futur.
- Prévision d'un état futur à partir d'un état connu.
- Espérance d'avoir d jours consécutifs ayant le même temps.
- etc.

Cependant, un modèle Markovien dans lequel chaque état correspond à un événement observable est trop restrictif pour être généralisé à un grand nombre de problèmes où les états ne sont pas directement observables. Dans la suite, nous allons donc nous intéresser aux modèles de Markov cachés.

3. Modèles de Markov cachés

Depuis 1975, les modèles de Markov cachés sont utilisés dans de nombreuses applications, principalement dans le domaine de la parole. Ces applications ne se contentent pas de s'appuyer sur la théorie des modèles de Markov cachés, mais développent plusieurs extensions théoriques dans le but d'améliorer les modèles. C'est ce qui a fait leur succès.

3.1 Définition générale

Un modèle de Markov caché est un processus doublement stochastique dont une composante est une chaîne de Markov non observable. Ce processus peut être observé à travers un autre ensemble de processus qui produit une suite d'observations. Plus simplement,

c'est un modèle qui décrit les états d'un processus markovien à l'aide des probabilités de transition et des probabilités d'observation par états [29]. Lors de la création d'un modèle de Markov caché, il existe 3 problèmes à résoudre : la reconnaissance, l'analyse et l'apprentissage. Ces problèmes seront analysés dans la suite. Afin de bien les appréhender, voici un exemple qui permet de voir l'application des solutions préconisées. L'exemple a pour objectif de définir à quelle saison nous sommes. Posons :

- Les 4 saisons (Printemps, Eté, Automne, Hiver), comme étant les états de notre modèle. L'état saison n'est pas directement observable mais il émet des observations du temps de la journée. Il est défini par Nuage, Pluie et Soleil.

- Une chaîne d'observation du temps de la semaine. Dans les exemples, cette chaîne est réduite à 3 jours pour limiter le nombre de calcul (Soleil, Soleil, Nuage) ou (S,S,N).

Le Modèle de Markov nous permettra de définir la chaîne de Markov qui a le plus de probabilité d'avoir généré la séquence observée, par exemple Eté, Eté, Printemps (E,E,P). La représentation graphique de ce modèle est donnée par la figure 3.1.

Dans cette figure, Nuage, Pluie et Soleil sont respectivement dénotés par N, P et S. A chaque nœud est associé la probabilité d'émission des états définis par Nuage, Pluie et Soleil.

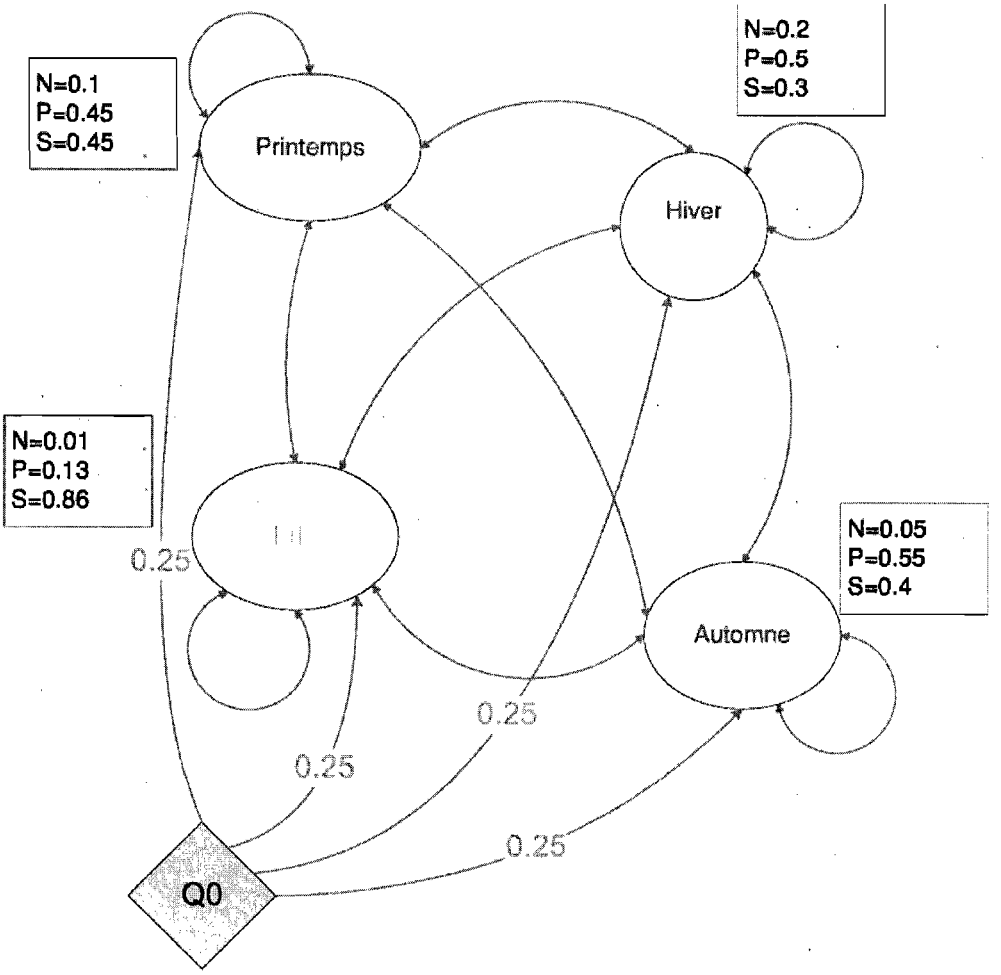


Figure 3.1 : Exemple de modélisation d'un modèle de Markov caché.

3.2 Apprentissage des modèles de Markov cachés

3.2.1 Présentation des modèles

Pour chaque mot dans la base de données, il faut définir un modèle prototype contenant la topologie choisie : le nombre d'états du modèle, les transitions possibles, la loi de probabilité associée à chaque état. Dans notre cas, une telle base spécifiera les types grammaticaux associés à un mot donné. Notons que les états initial et final ont la particularité de ne pas émettre d'observations, mais de servir uniquement à la connexion des modèles de parole continue, les modèles peuvent avoir des prototypes différents. Les probabilités d'émission correspondent à des modèles de Markov cachés, caractérisés par leur moyenne et leur matrice de covariance.

3.2.2 Étiquetage

Avant l'apprentissage des modèles, il est nécessaire de préparer les données d'apprentissage en calculant les paramètres du signal d'une part et en étiquetant les phrases d'apprentissage d'autre part. Ces phrases doivent, bien sûr, être étiquetées en fonction des unités modélisées : il faut que tous les segments correspondent aux unités. Cette tâche fastidieuse (segmentation manuelle en général) peut être limitée à une fraction de l'ensemble d'apprentissage. Il faut donc parcourir un fichier de données, calculer la moyenne et la variance et fixer les valeurs des lois gaussiennes des modèles de Markov cachés.

3.2.3 Reconnaissance

Un module de décodage d'image doit utiliser l'algorithme de Viterbi pour trouver la séquence d'états la plus probable correspondant aux paramètres observés dans un modèle composite, et en déduire les unités correspondantes. Le modèle composite autorise la succession des modèles en fonction d'une syntaxe choisie préalablement. Celle-ci peut se situer au niveau phonétique ou lexical : les mots du lexique peuvent être définis par la concaténation d'unités sub-lexicales.

3.3 Choix d'une Topologie

Le choix de la topologie influe la qualité de la reconnaissance. En effet, c'est suite à ce choix que seront définis le nombre d'états du modèle, les transitions possibles ainsi que la loi de probabilité associée à chaque état.

3.3.1 Topologie ergodique

Tout état peut être atteint depuis tout autre état en un nombre fini de transitions. Ce type de modèle est plus général et intéressant lorsque le modèle représente un processus dont nous voulons suivre les évolutions des états. La figure 3.2 donne un exemple d'une topologie ergodique.

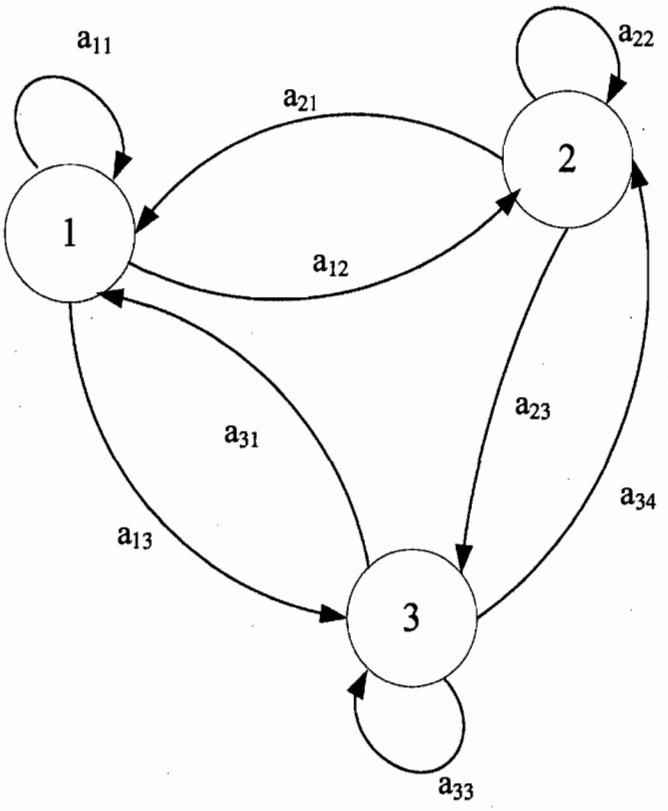


Figure 3.2: Topologie ergodique.

Figure 3.2 montre que les états spécifiés par 1, 2, et 3 sont atteignables à partir de tous les autres états. On note ainsi 3 arcs entrant pour chacun d'eux et 3 sortant dont un réflexif (a_{11} pour l'état 1).

.3.2 Topologie gauche droite

Elle est utilisée pour suivre des observations dont l'évolution se fait dans un ordre donné telle que la reconnaissance de l'écriture (*cf.* Figure 3.3)

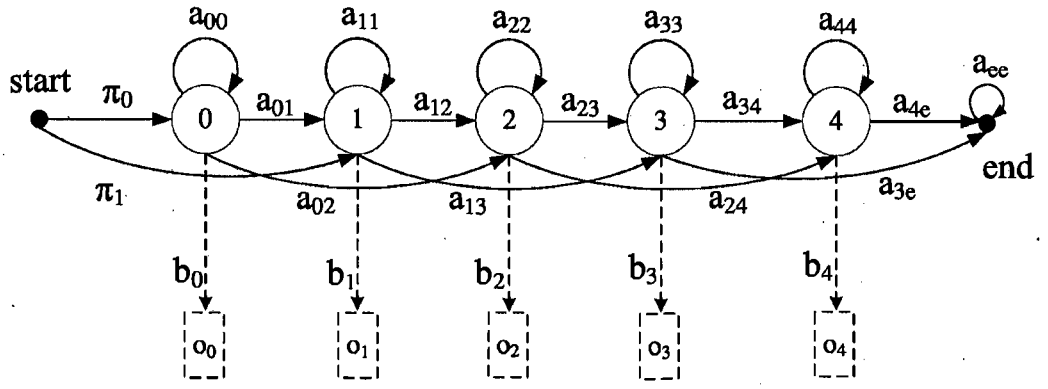


Figure 3.3 : Topologie gauche droite.

Formellement : $a_{ij} = 0$ si $j < i$; $\pi_i = 0$ si $i \neq 1$ et $\pi_i = 1$ si $i = 1$. De même, on a souvent des contraintes supplémentaires comme : $a_{ij} = 0$ si $j > i + \Delta$ (par exemple $\Delta = 2$ dans le modèle de Bakis). Ces modèles permettent de modéliser des signaux qui évoluent avec le temps (par exemple, la parole).

3.4 Historique et champs d'applications

Voici l'historique de l'évolution des recherches sur les chaînes de Markov (caches) et leurs applications dans différents domaines [30] :

- 1913 : La théorie des chaînes de Markov. Une première application a été développée par Markov pour analyser le langage [27].

- 1948/1951 : Les travaux de Shannon concernant la théorie de l'information utilisant les modèles de Markov [31,32].

- 1957 : Les travaux de Bellman concernant la programmation dynamique [33].

- 1958/1961 : Les premières applications : les modèles probabilistes d'urnes, le calcul direct du maximum de vraisemblance, et l'observation de la suite d'états dans une chaîne de Markov.

• 1966/1974 : Des recherches ont abouti à développer des algorithmes efficaces pour l'estimation des paramètres du modèle de Markov caché et pour le décodage de la suite d'états cachés. A cet effet, des algorithmes itératifs, basés sur le maximum de vraisemblance, ont été développés pour l'estimation des états cachés et l'estimation des paramètres du modèle. Des généralisations permettent d'inclure la notion de durée variable et des densités de probabilité continues multi variables ont été développés. Viterbi a écrit son algorithme, qui possède une complexité linéaire avec la longueur de la suite d'observations à décoder et qui permet d'estimer la suite des états du modèle correspondant au meilleur chemin. Ces algorithmes ont été largement exploités au décodage de la parole. Jelinek conçoit son algorithme qui réalise une exploration arborescente des chemins possibles en utilisant une pile [34]. En 1970, l'utilisation des modèles de Markov cachés devient plus nette grâce à leur description par l'exemple des urnes de la part de Cave et Neuwirch qui ont employé pour la première fois l'expression « Hidden Markov models » au lieu de « Probabilistic function of a Markov chain » [28].

• 1975/1993 : Les modèles de Markov cachés sont utilisés dans de nombreuses applications, principalement dans le domaine de la parole. Mais ces applications ne se contentent pas de s'appuyer sur la théorie des modèles de Markov cachés, ils développent également plusieurs extensions théoriques dans le but d'améliorer les modèles.

Les modèles de Markov cachés ont prouvé dans de nombreux domaines qu'ils étaient de outils puissants.

4. Les trois problèmes fondamentaux des modèles de Markov cachés

Voici la liste des problèmes qu'il est nécessaire de résoudre afin de pouvoir utiliser les modèles de Markov cachés lors de la modélisation d'un processus réel.

1. Reconnaissance : Etant donnés un modèle de Markov caché $\lambda = \{\Pi, A, B\}$ et une séquence observée $O = \{o_1, o_2, ..., o_n\}$, quelle est la vraisemblance $P(O/\lambda)$ que le modèle λ génère O ?

2. Analyse : Etant donnés un modèle de Markov caché λ et une séquence observée O, quelle est la séquence des états qui a la probabilité maximale d'avoir généré O?

3. Apprentissage : A partir d'une chaîne d'observations O = $\{o_1, o_2, ..., o_n\}$, comment ajuster les paramètres du modèle de Markov caché λ = {Π, A, B} pour maximiser la vraisemblance de l'ensemble d'apprentissage $P(O/\lambda)$?

4.1 Problème 1 : Reconnaissance

Etant donnés une suite d'observations $O = \{o_1, o_2, ..., o_n\}$ et un modèle, comment peut-on calculer efficacement la probabilité (vraisemblance) pour que la suite d'observations O soit produite par λ, c'est-à-dire $P(O/\lambda)$? Autrement dit, comment évaluer le modèle afin de choisir parmi plusieurs celui qui génère le mieux la suite d'observations? Plusieurs techniques permettent de résoudre ce problème : méthode d'évaluation directe, procédure « Forward-Backward » et Algorithme de Viterbi.

4.1.1 Evaluation directe

La probabilité $P(O/\lambda)$ d'une suite d'observations $O = \{o_1, o_2, ..., o_n\}$, connaissant le modèle λ = {Π, A, B}, est la somme sur tous les chemins de la probabilité que le chemin $Q = \{q_1, q_2, ..., q_T\}$ en cours ait généré l'observation, soit :

$$P(O/\lambda) = \sum_{Q} P(O/Q,\lambda)P(Q/\lambda). \tag{3.5}$$

La définition de la probabilité d'emprunter le chemin Q est comme suit :

$$P(Q/\lambda) = P(s_1, s_2, s_3..., s_n/\lambda) = \pi_1 * a_{1,2} * a_{2,3} * ... * a_{n-1,n}. \tag{3.6}$$

La probabilité que cette séquence Q émette les signaux observations O est comme suit :

$$P(O/Q,\lambda) = P(O/s_1, s_2, s_3..., s_n, \lambda) = b_1(o_1) * b_2(o_2) * ... * bn(o_n). \tag{3.7}$$

Nous obtenons :

$$P(O/\lambda) = \sum_{Q} P(\pi_1) * a_{1,2} * b_1(a_1) * a_{2,3} * b_2(o_2) * ... * a_{n-1,n} * b_n(o_n). \tag{3.8}$$

Avec a_{ij} étant la valeur de la cellule relative à l'état s_i et l'état s_j dans la matrice de transition A, b_i étant une observation faisant partie de la matrice d'observation B et π_i étant la probabilité d'initialisation de l'état s_i.

Ceci générera $(2T - 1) * N^T$ *multiplications et* N^T *additions*, soit environ $2T*N^T$ opérations. Par exemple, si N = 5 (états), T = 100 (observations), alors nous devons faire de l'ordre de $(2 * 100 * 5100) = 10^{72}$ opérations Cette solution n'est pas acceptable car à 1Ghz, en supposant qu'une opération de calcul coûte une opération élémentaire du CPU, la réponse sera trouvée au bout de $3*10^5$ années.

Nous constatons que de nombreuses multiplications sont répétées (portions de sous séquences communes). L'idée est alors de calculer $P(O/\lambda)$ de manière incrémentale : L'algorithme Forward-Backward.

4.1.2 Algorithme Forward-Backward

Dans cette approche, nous considérons que l'observation peut se faire en deux étapes :

- L'émission de la suite d'observations $O = \{o_1, o_2, ..., o_t\}$ et la réalisation de l'état q_t au temps t : Forward.
- L'émission de la suite d'observations $O = \{o_{t+1}, o_{t+2}, ..., o_T\}$ en partant de l'état q_i au temps t : Backward.

$P(O/\lambda)$ peut être défini à chaque instant $t \in [1,T]$ par :

$$P(O/\lambda) = \sum_{i=1}^{N} \alpha_t(i) * \beta_t(i) = 1. \qquad (3.8)$$

Où $\alpha_t(i)$ est la probabilité d'émettre la suite $\{o_1, o_2, ..., o_t\}$ et d'aboutir à q_t à l'instant t et $\beta_t(i)$ est la probabilité d'émettre la suite $\{o_{t+1}, o_{t+2}, ..., o_T\}$ en partant de l'état q_t au temps t, connaissant λ. Le calcul de $\alpha_t(i)$ se fait avec t croissant tandis que celui de $\beta_t(i)$ se fait avec t décroissant, d'où l'expression Forward-Backward.

(a) La variable Forward

Soit la probabilité $\alpha_t(i) = P(O, q_t = s_i/\lambda)$ pour générer $O = \{o_1, o_2, ..., o_n\}$ et de se trouver dans l'état q_t à l'instant t. Un exemple de la variable Forward est donné par la figure 3.4.

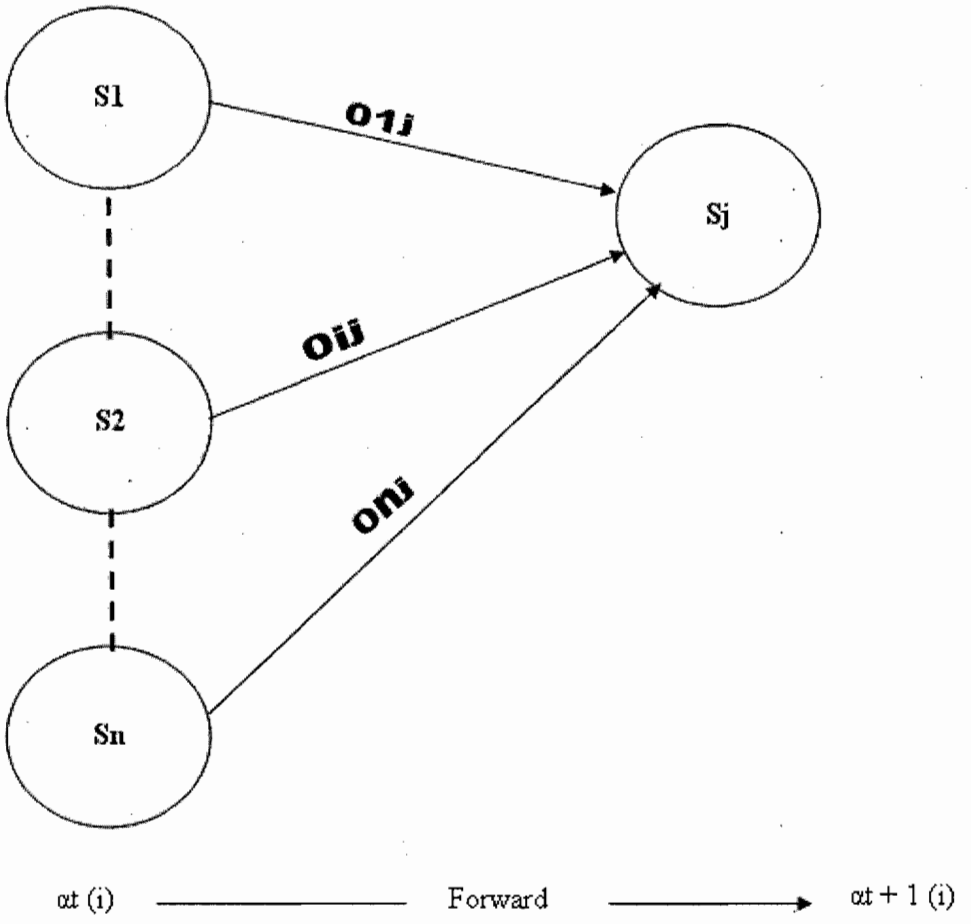


Figure 3.4 : Exemple de la variable Forward.

Cet algorithme permet de calculer cette probabilité :

Algorithme 3.1 : Algorithme Forward

---

1. Initialisation: $\alpha_1(i) = \Pi_i * b_i(o_1)$.
2. Itérations: $\alpha_{t+1}(j) = [\sum_{i=1}^{N} \alpha_t(i) * a_{ij}] * b_j(o_{t+1}),\ pour\ t \in [1, T-1],\ j \in [1, N]\Pi_i * b_i(o1)$
3. Terminisation: $P(O/\lambda) = \sum_{i=1}^{N} \alpha_t(i)$.

Ceci génèrera N*(N+1)*(T-1) + N multiplications et N*(N-1)*(T-1) additions, soit environ 2 * N^2 * T opérations. Par exemple, si N=5 (états) et T=100 (observations), alors nous devons faire de l'ordre de 2 * 5^2 * 100 $\approx$ 5000 opérations.

Le Forward répond au problème 1.

Le Backward est développé ici, mais il sera utilisé pour résoudre le problème 3.

(b) La variable Backward

Soit la probabilité $\beta_t(i) = P(O/q_t = s_i, \lambda)$ de générer O = $\{o_{t+1}, o_{t+2}, ..., o_T\}$ sachant que nous l'avons dans l'état q_t à l'instant t. Un exemple de la variable Backward est donné par la figure 3.5.

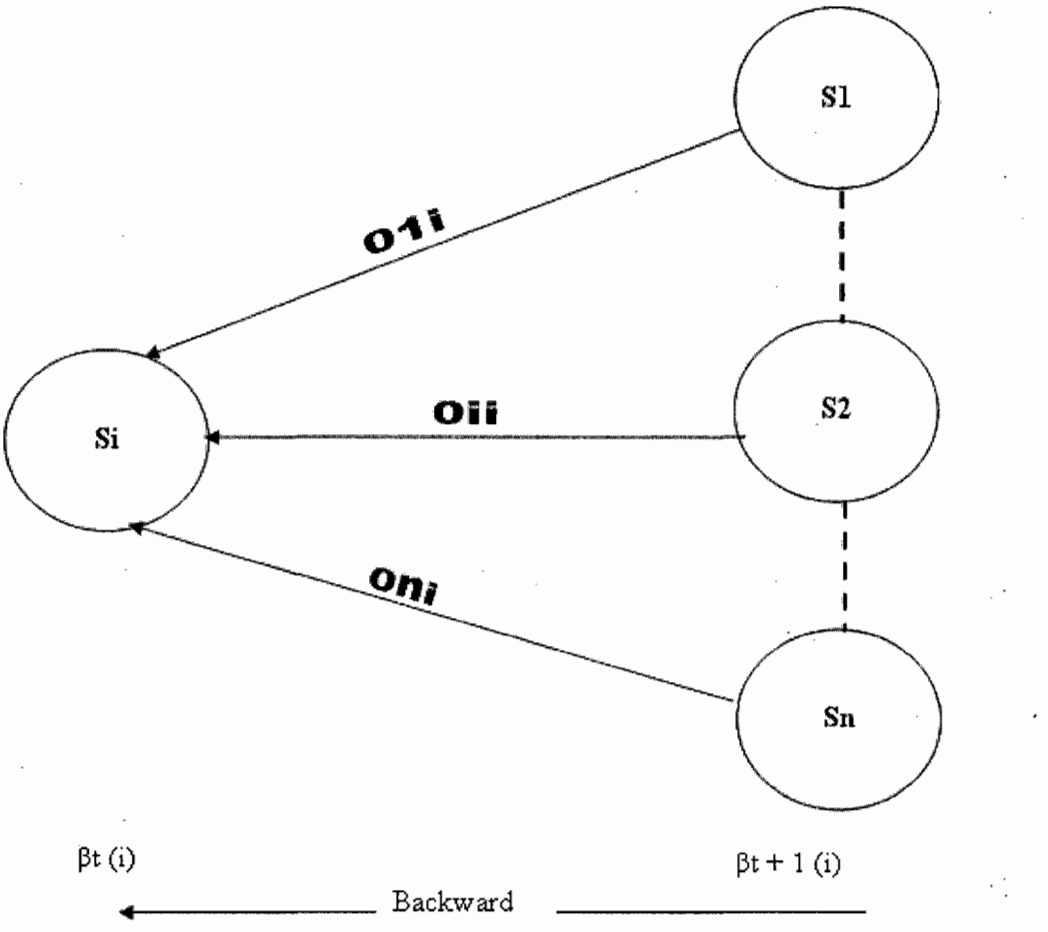


Figure 3.5 : Exemple de variable Backward.

Elle représente la probabilité qu'un modèle λ aie pu générer la tranche de séquence $o_{t+1},\ldots,o_T$ sachant que ce modèle se trouvait dans l'état s_i à l'instant t. Le calcul de ce vecteur de variable

pour $1 \leq t \leq T$ et $1 \leq i \leq N$ se fait d'une façon similaire à celle établie pour les variables $\alpha_t(i)$ sauf que la progression se fait dans le sens inverse de l'axe temporelle.

Cet algorithme permet de calculer cette probabilité :

Algorithme 3.2 : Algorithme Backward

---

1. Initialisation: $\beta_T(i) = 1, 1 \leq i \leq N.$
2. Itérations: $\beta_t(i) = [\sum_{j=1}^{N} a_{ij} * b_j(o_{t+1})\beta_{t+1}(j)],\ pour\ t \in [T-1, 1], j \in [1, N]$

Pour être dans l'état q_t à l'instant t, tout en tenant compte de la suite d'observations de O = $\{o_{t+1}, o_{t+2}, ..., o_T\}$, il faut considérer tous les états possibles s_j (toutes les transitions a_{ij}) et l'observation o_{t+1} dans l'état j (les $b_j(o_t+1)$ puis la suite d'observations partielle restante à partir de l'état j $(\beta_{t+1}(j))$.

Ceci générera N*(N +1)*(T-1) + N multiplications et N*(N-1)*(T-1) additions, soit environ 2 * N^2 * T opérations. Par exemple, si N=5 (états) et T=100 (observations), alors nous devons faire de l'ordre de 2 * 5^2 * 100 $\approx$ 5000 opérations.

4.2 Problème 2 : Analyse

Etant donnés une suite d'observations $O = \{o_1, o_2, ..., o_n\}$ et un modèle λ = {Pi, A, B}, comment peut-on choisir une suite d'états Q = $\{q_1, q_2, ..., q_n\}$ qui soit optimale selon un critère convenable? La difficulté réside dans la suite optimale d'états, il existe plusieurs méthodes : le critère local, le critère global et l'algorithme de Viterbi.

D'un point de vue mathématique, nous cherchons la séquence qui a la plus grande probabilité d'avoir générée O et ce parmi M^n possibilités :

$$\max P(O, I/\lambda) * I, \text{ parmi } M^n \text{ possibilités.}$$

La figure 3.6 montre le lien qui existe entre les observations et les séquences.

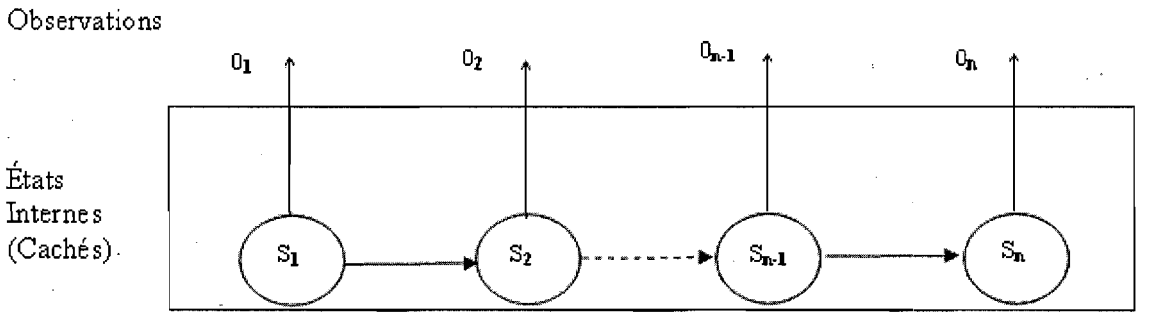


Figure 3.6 : Lien observations séquences.

4.2.1. Critère local pour l'obtention d'une séquence optimale d'états

Cette technique consiste à choisir l'état q_t qui est le plus probable et ceci indépendamment des autres états individuellement pour chaque t. Soit la probabilité $\gamma_t(i)$ d'être dans l'état s_i au temps t.

$$\gamma_t(i) = P(q_t = s_i / O, \lambda) alors : \gamma_t(i) = \frac{P(q_t = s_i / O, \lambda)}{P(O, \lambda)} = \frac{\alpha_t(i) * \beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i) * \beta_t(i)} \tag{3.9}$$

Notons que nous pouvons calculer γ_t une fois α_t et β_t calculés, de plus : $\sum_{i=1}^{N} \gamma_t(i)$

Nous pouvons résoudre cette équation par : argmax[γ_t(i)] pour tout t $\in$ [1, T] et pour tout i $\in$ [1, N]. Néanmoins, cette méthode n'est pas viable car rien ne garantit que les transitions entre chaque état de Q_t soient valides. Soit l'exemple illustré par la figure 3.7.

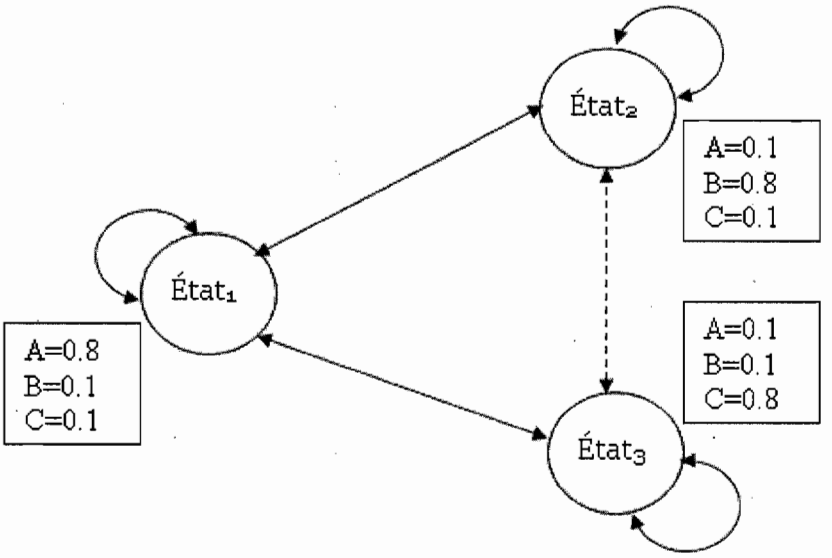


Figure 3.7 : Exemple de transitions non valides.

Selon ce critère, l'observation ABC aura été générée par la chaîne de Markov 123, (l'état 1 a la plus grande probabilité de générer A). Pourtant, la probabilité de passer de l'état 2 à l'état 3 est de 0. Il est donc impossible d'obtenir une transition entre ces deux états. Ceci est dû au calcul du « argmax » qui détermine l'état le plus vraisemblable à chaque instant sans prendre en compte la probabilité d'occurrences des suites d'états.

4.2.2. Critère global pour l'obtention d'une séquence optimale d'état

Nous cherchons à trouver l'unique trajectoire optimale de la suite d'états, donc de maximiser P(Q/O,λ). Pour cela, nous pouvons définir la probabilité maximale d'une séquence au temps t qui se termine dans l'état q_t.

En conservant pour chaque t et chaque i l'état ayant amené au maximum : $\gamma_t(j)$, $\psi_t(j)$, nous obtenons l'algorithme de Viterbi.

4.2.3 Algorithme Viterbi

Cette solution est la plus utilisée. Elle est basée sur les techniques de programmation dynamique. C'est un algorithme récursif qui permet de trouver à partir d'une suite d'observations, une solution optimale au problème d'estimation de la suite d'états.

Algorithme 3.3 : Algorithme Viterbi

1. *Initialisation* : *For* $1 \leq i \leq N \delta_1(i) = \pi_i * b_i(o_1), \psi_i(i) = 0$
2. Calcul itteratif: *For* $2 \leq t \leq T$
   For $1 \leq j \leq N$
   $\delta_t(j) = \max_i [\delta_{t-1}(i) * a_{ij}] b_i(O_t)$
   $\psi_t(j) = argmax[\delta_{t-1}(i) * a_{ij}]$
3. Calcul itteratif: $P^* = \max_i [\delta_T(i)]$
   $Q_T^* = argmax_i [\delta_{t-1}(i) * a_{ij}]$
4. Backtracking: *For* $t = T - 1$ *to*1
   $q * t = \psi_t(q_t^*$

Il est important de noter qu'hormis l'étape du backtracking, l'algorithme de Viterbi est similaire au calcul du Forward. Dans cet algorithme, nous utilisons le « argmax » des probabilités au lieu de la somme. Cet algorithme nécessite N cases mémoire, une case par état permettant de stocker $\gamma_t(i)$, $\psi_t(q_t)$ de longueurs T symboles. P* est la probabilité la plus élevée d'une séquence expliquant O. La séquence associée (il peut en avoir plusieurs) se trouve en « backtrackant » (en suivant les pointeurs arrière ψ). La figure 3.8 donne un exemple illustrant l'algorithme 3.3.

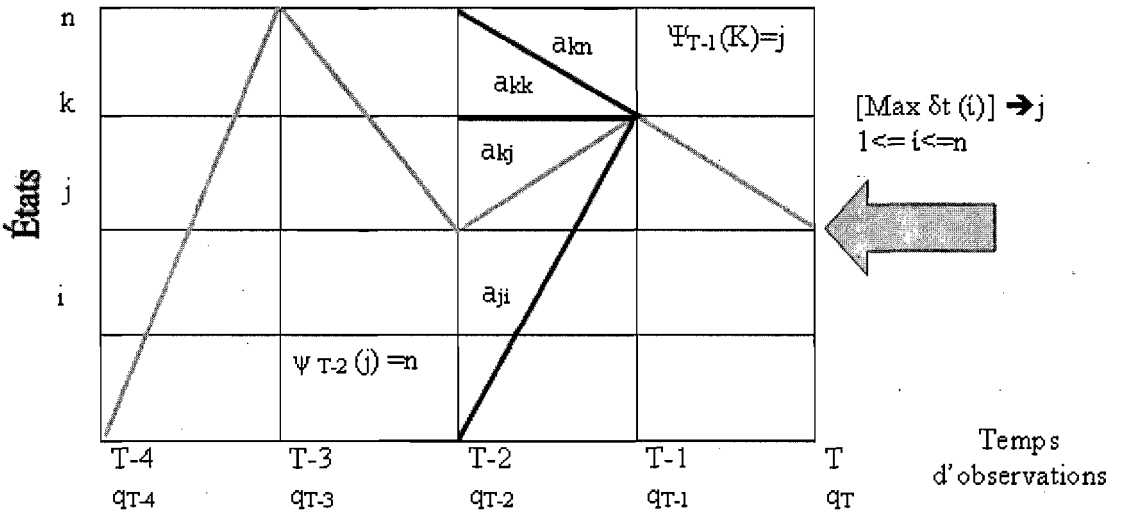


Figure 3.8 : Le backtracking.

4.3 Problème 3 : Apprentissage

Comment peut-on ajuster les paramètres du modèle $\lambda = \{\Pi, A, B\}$ pour maximiser $P(O/\lambda)$? Le fait que la longueur de la suite d'observations (données d'apprentissage) soit finie implique l'absence de solutions analytiques directes (d'optimisation globale) pour construire le modèle. Néanmoins, nous pouvons choisir $\lambda = \{\Pi, A, B\}$ tel que $P(O_t/\lambda)$ est un maximum local en utilisant une procédure itérative telle que celle de Baum-Welch. L'idée de l'application est donc d'utiliser des procédures de re-estimation qui affinent le modèle, petit à petit, selon les étapes suivantes :

- Choisir un ensemble initial de paramètres λ_0.
- Calculer λ_1 partir de λ_0.
- Répéter ce processus jusqu'à un critère de fin.

La probabilité de passer par s_i en t et s_j en t+1 en générant O avec λ :

$$\xi_t(i,j) = P(q_t = s_i, q_{t+1} = s_j / O, \lambda) \tag{3.10}$$

Les variables backward et forward nous permettent d'écrire :

$$\frac{P(q_t = s_i, q_{t+1} = s_j / O, \lambda)}{P(O/\lambda)} \tag{3.11}$$

D'où :

$$\frac{\alpha_t(i) * a_{i,j} * b_j(O_{t+1}) * \beta_{t+1}(j)}{P(O/\lambda)} \tag{3.12}$$

Si $\gamma_t(i)$ est le nombre espéré de transitions de s_i vers s_j, nous pouvons l'écrire :

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j), \ pour\ t \in [1, T-1] \tag{3.13}$$

L'objectif est d'obtenir la séquence d'opérations nécessaires pour que le système soit à l'état s_i au temps t et à l'état s_j au temps t+1.

$$\gamma_i j = \sum_{i=1}^{T-1} \xi_t(i,j) \tag{3.14}$$

La figure 3.9 donne un exemple de séquence d'opérations.

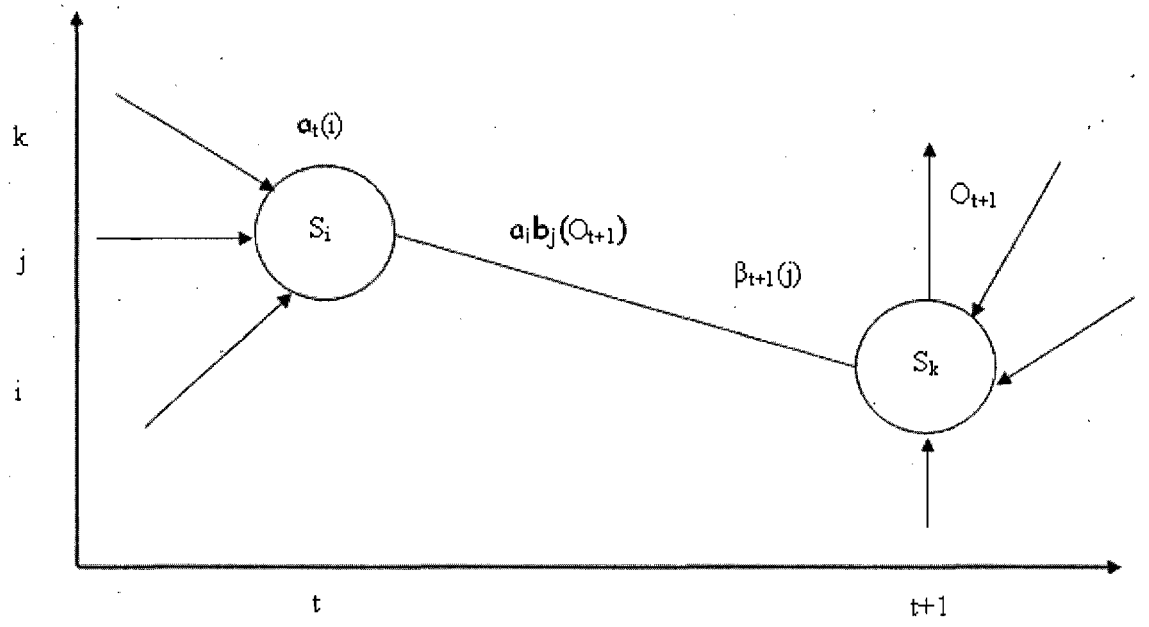


Figure 3.9 : Séquence d'opérations pour Baum-Welch.

Cette méthode de maximum de vraisemblance est la plus utilisée dans les applications. Cet algorithme peut être représenté sous la forme itérative suivante :

Algorithme 3.4 : Algorithme Baum-Welch

---

1. Fixer des valeurs initiales k = 0
   $\lambda = \{\pi_i^0, a_i j^0, b_j^0(O_t)\}$; *pour* $i \in [1;N]$ *et* $j \in [1;N]$
2. Calculer, en utilisant les variables Forward Backward : $\xi_t(i,j)$ *et* $\gamma_t(i)$; *pour* $i \in [1;N]$ *et* $j \in [1;N]$
3. Nouvelles estimations k = 1,2,3... $\lambda = \{\pi_i^k, a_i j^k, b_j^k(O_t)\}$; *pour* $i \in [1;N]$ *et* $j \in [1;N]$
4. Recommencer en 2 jusqu'à un certain point limite, défini ci-dessous.

Le test d'arrêt est généralement un nombre d'itérations qui est fixé empiriquement. Le choix d'un modèle initial influe sur les résultats : toutes les valeurs nulles de A et de B au

départ, restent à zéro à la fin de l'apprentissage. Il est à noter que l'algorithme converge vers des valeurs de paramètres qui forment un point critique de $P(O_t/\lambda)$. Donc, nous obtenons un maximum local ou un point d'inflexion. D'ou la nécessité de bien choisir le modèle initial. Pour avoir une estimation convenable du modèle, les « re-estimations » se font sur un ensemble de plusieurs suites d'observations appelées corpus d'apprentissage. Donc la taille du corpus d'apprentissage influe, elle aussi, sur les résultats. Il est souhaitable que celle-ci soit importante.

5. Conclusion

Dans ce chapitre, nous avons donné une présentation formelle des modèles de Markov observables et cachés. Ensuite, nous nous somme concentrés sur les modèles de Markov caché. A cet effet, nous avons présenté les caractéristiques de ce type de modèles et ensuite, nous avons détaillé les trois principaux problèmes des modèles de Markov cachés.

Chapitre 4 : Réseaux bayésiens

1. Introduction

Les réseaux bayésiens font partie de la famille des modèles graphiques. Ils regroupent au sein d'un même formalisme la théorie des graphes et celle des probabilités afin de fournir des outils efficaces autant qu'intuitifs pour représenter une distribution de probabilités jointe sur un ensemble de variables aléatoires.

Ce formalisme très puissant permet une représentation intuitive de la connaissance sur un domaine d'application donné et facilite la mise en place de modèles performants et clairs. La représentation de la connaissance se base sur la description, par des graphes, des relations de causalité existant entre des variables décrivant le domaine d'étude. A chaque variable est associée une distribution de probabilité locale quantifiant la relation causale.

Dans ce chapitre, nous allons commencer par donner une présentation sommaire des modèles graphiques. Ensuite, nous intéresserons plus en détails aux réseaux bayésiens.

En effet, dans un premier temps, nous allons donner une définition intuitive de cette technique, et ensuite nous allons enchaîner par sa définition formelle. Dans un second temps, nous allons présenter le formalisme des réseaux bayésiens en abordant aussi les notions d'indépendance conditionnelle et de d-séparation.

2. Les modèles graphiques

Les modèles graphiques portent de nombreux noms : réseaux de croyance, réseaux probabilistes, réseaux d'indépendance probabiliste ou encore réseaux bayésiens. Il s'agit d'un formalisme pour représenter de façon factorisée une distribution jointe de probabilités sur un ensemble de variables aléatoires. Ils ont révolutionné le développement des systèmes intelligents dans de nombreux domaines. En effet, ils forment l'association entre la théorie des probabilités et la théorie des graphes. Ils apportent des outils naturels permettant de traiter deux grands problèmes couramment rencontrés en intelligence artificielle, en mathématiques appliquées ou en ingénierie : l'incertitude et la complexité. Ils jouent en particulier un rôle croissant dans la conception et l'analyse d'algorithmes liés au raisonnement ou à l'apprentissage.

A la base des modèles graphiques se trouve la notion fondamentale de la modularité : un système complexe est construit par la combinaison de parties simples. La théorie des probabilités fournit le ciment permettant la combinaison de ces parties, tout en assurant que le modèle est et reste consistant. Elle fournit alors un moyen d'interfacer modèles et données.

La théorie des graphes apporte d'une part une interface intuitive grâce à laquelle un humain peut modéliser un problème comportant des variables entre lesquelles existent des interactions, et d'autre part un moyen de structurer les données. Ceci conduit alors vers une conception naturelle d'algorithmes génériques efficaces.

Beaucoup de systèmes probabilistes classiques issus de domaines tels que les statistiques, la théorie de l'information, la reconnaissance des formes ou encore la mécanique statistique, sont en fait des cas particuliers du formalisme plus général que constituent les modèles graphiques. Parmi ces modèles, nous pouvons citer les modèles de Markov, les arbres de décision ou encore les modèles d'Ising. Ainsi, les modèles graphiques sont un moyen efficace de voir tous ces systèmes comme des instances d'un formalisme commun sous-jacent.

L'avantage immédiat réside dans le fait que les techniques développées pour certains domaines peuvent alors être aisément transférées à un autre domaine et être exploitées plus facilement. Les modèles graphiques fournissent ainsi un formalisme naturel pour la conception de nouveaux systèmes.

Dans la section suivante, nous nous intéressons plus en détails aux réseaux bayésiens.

3. Réseaux bayésiens

3.1 Introduction

Dans cette section, nous allons nous focaliser sur un modèle particulier de la famille des modèles graphiques : les réseaux bayésiens. Ces derniers utilisent des graphes acycliques dirigés (GAD). Le rôle de ces graphes dans les modèles probabilistes et statistiques est triple :

1. fournir un moyen efficace d'exprimer des hypothèses,

2. donner une représentation économique des fonctions de probabilité jointe,

3. faciliter l'inférence à partir d'observations.

Supposons que nous disposions d'un ensemble U de variables aléatoires discrètes tel que U = $\{x_1, x_2, ..., x_n\}$. Pour être stockée, la probabilité jointe P(U) de cet ensemble nécessitera un

tableau comprenant 2^n entrées : une taille particulièrement grande, quelque soit le système utilisé. Par contre, si nous savons que certaines variables ne dépendent en fait que d'un certain nombre d'autres variables, alors nous pouvons faire une économie substantielle en mémoire et par conséquent en temps de traitement. De telles dépendances vont nous permettre de décomposer cette très large distribution en un ensemble de distributions locales beaucoup plus petites, chacune ne s'intéressant qu'à un petit nombre de variables. Les dépendances vont aussi nous permettre de relier ces petites distributions en un grand ensemble décrivant le problème que nous voulions modéliser. Nous pourrons ainsi répondre de façon cohérente et efficace à diverses questions que nous pourrions nous poser sur cette distribution de probabilité.

Dans un graphe, il est possible de représenter chaque variable du problème par un noeud et chaque dépendance entre les variables par un arc.

Les graphes dirigés et non dirigés [21] sont abondamment utilisés pour faciliter une telle décomposition des connaissances. Les graphes acycliques dirigés sont utilisés pour représenter des relations temporelles ou causales. Judea Pearl les a nommés Réseaux Bayésiens en 1985 pour mettre en évidence trois aspects [35] :

1. la nature subjective des informations,

2. l'utilisation de la formule de Bayes comme principe de base pour la mise à jour des informations,

3. la distinction entre les modes de raisonnement causal et fondé (basé sur des évidences).

3.2 Définition d'un réseau bayésien

3.2.1 Définition intuitive d'un réseau bayésien

Plusieurs définitions intuitives ont été attribuées aux réseaux bayésiens parmi lesquelles nous citons :

Définition :

" Un réseau bayésien (ou probabiliste ou de croyance) est un graphe dont les noeuds définissent les variables du système et dont les arcs définissent l'existence de relations entre ces variables... Un réseau bayésien est en quelque sorte un réseau causal, auquel nous ajoutons des éléments de probabilités issus de la théorie bayésienne. Ce qui est important avec les réseaux bayésiens, c'est le fait que les relations causales ne sont pas absolues, mais sont

associées à une probabilité indiquant le degré de croyance que nous avons dans l'évènement." [36]

En fait, un réseau bayésien est un graphe orienté, acyclique, constitué de noeuds et d'arcs. Les noeuds représentent des variables généralement discrètes. Néanmoins, il y a quelques extensions aux réseaux bayésiens pour représenter des variables continues. Les arcs sont orientés et représentent des liens de dépendance directe (de causalité) entres les noeuds, l'absence d'arc ne renseigne que sur l'inexistence d'une dépendance directe. Certains noeuds n'ont pas de noeuds parents (les variables qu'ils représentent ne dépendent d'aucune autre variable) : ils sont appelés *noeuds racines*. Le grand avantage des réseaux bayésiens est de permettre de modéliser des relations non déterministes [35]. Ceci grâce au concept de **table des probabilités conditionnelles** associée à chaque noeud.

Afin de mieux comprendre ce qu'est qu'un réseau bayésien, nous empruntons l'exemple simple mais efficace utilisé dans [35]. Cet exemple représente le problème des retards à l'école. Il est supposé alors que le retard d'un élève peut être causé par un mauvais état de santé ou par un retard du bus, alors que le retard d'un enseignant ne peut être causé que par un mauvais état de santé. Le réseau bayésien correspondant est donné par la figure 4.1.

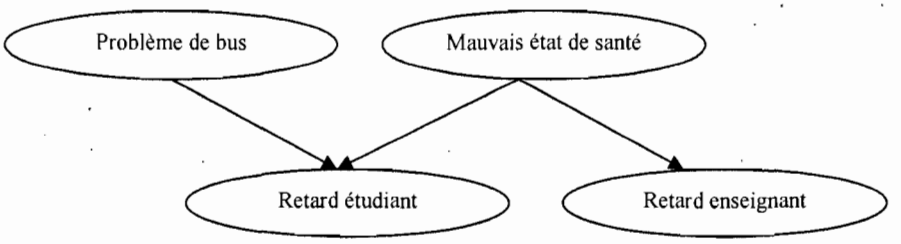


Figure 4.1 : Un réseau bayésien modélisant le problème des retards à l'école.

Quatre variables sont représentées dans ce réseau :

- Variable 1 : Problème de bus ;
- Variable 2 : Mauvais état de santé ;
- Variable 3 : Retard étudiant ;
- Variable 4 : Retard enseignant.

Notons que les variables 1 et 2 sont représentées par des nœuds racines étant donné que ces nœuds n'ont pas de nœuds parents.

Ces quatre variables prennent seulement pour valeur soit *Vrai* soit *Faux*. La table 4.1 montre ce que pourrait être la table de probabilités conditionnelles associée au nœud *Retard enseignant*. Elle modélise la dépendance entre le retard de l'enseignant et son mauvais état de santé. Cette table est en fait la distribution de la probabilité de l'attribut *Retard enseignant*, conditionnelle à l'attribut *Mauvais état de santé*. Elle donne seulement la probabilité de l'événement *Retard enseignant* = *Vrai*, car P(*Retard enseignant* = *Faux*) = 1 − P(*Retard enseignant* = *Vrai*).

	mauvais état de santé = *Vrai*	mauvais état de santé = *Faux*
P(Retard enseignant = *Vrai*)	0,6	0,1

Table 4.1 : Table de probabilités conditionnelles relative à la variable *Retard enseignant*.

Les tables de probabilités associées respectivement aux noeuds *Mauvais état de santé* et *Retard du bus* ont une nature particulière. En effet, comme mentionné auparavant, ces noeuds n'ont pas de noeuds parents dans ce modèle, ce sont des *nœuds racines*, et se voient donc assignés des probabilités pour leurs deux valeurs *Vrai* et *Faux*, en supposant par exemple que P(*Mauvais état de santé* = *Vrai*) = 0,4 et P(*Retard du bus* = *Vrai*) = 0,2.

3.2.2 Définition formelle d'un réseau bayésien

Avec la représentation graphique de la causalité (*cf.* Figure 4.1), nous pouvons connaître la direction de circulation de connaissances dans le graphe. En effet, la relation entre la cause et l'effet est représentée par une flèche orientée telle que celle entre les variables Variable 1 et Variable 3. Cependant, nous ne pouvons pas connaître la quantité de cette circulation de connaissances. Ainsi, il faut une représentation probabiliste associée avec le graphe. C'est pour cette raison qu'à la relation causale **A → B** est associée la probabilité conditionnelle P(B|A) représentant la quantité de cette relation. Rappelons que P (B|A) signifie la probabilité d'avoir B ayant A.

Définition :

Un réseau bayésien peut être formellement défini comme suit :

- un graphe acyclique orienté G, G = G(V,E) où V est l'ensemble des nœuds de G, et E l'ensemble des arcs de G.

- un espace probabilisé fini (Ω, p).

- un ensemble de variables aléatoires associées aux noeuds du graphe et définies sur (Ω, p) tel que :

$$p(V_1, V_2, ..., V_n) = \prod_{i=1..n} p(V_i \mid C(V_i)) \qquad (4.1)$$

avec $C(V_i)$ est l'ensemble des parents de V_i dans le graphe G (en d'autres termes, $C(V_i)$ est l'ensemble des causes de V_i dans le graphe G)

Un réseau bayésien est donc un graphe causal auquel nous avons associé une représentation probabiliste sous-jacente. Cette représentation permet de rendre quantitatifs les raisonnements sur les causalités que nous pouvons faire à l'intérieur du graphe.

3.4 Indépendance conditionnelle et d-séparation

Considérons trois ensembles disjoints de variables X, Y et Z représentés par trois ensembles de noeuds dans un graphe acyclique dirigé G. Pour savoir si X est indépendant de Y sachant Z dans toute distribution compatible avec G, nous avons besoin de tester si des noeuds correspondants aux variables de Z bloquent tous les chemins allant des noeuds de X vers les noeuds de Y.

Un chemin est une séquence consécutive d'arcs (non dirigés) dans le graphe. Un blocage peut être vu comme un arrêt du flux d'informations entre les variables qui sont ainsi connectées. Le flux d'information est dirigé par le sens des arcs et représente le flux des causalités dans le graphe ou l'ordre dans lequel les influences vont se propager dans le graphe. Cette propagation des influences peut alors être vue comme un envoi d'information d'une variable à ses variables filles.

La d-séparation[4] permet ainsi de déterminer si deux variables quelconques sont indépendantes conditionnellement à un ensemble de variables instantiées.

Définition : d-séparation

Un chemin *p* est dit d-séparé (ou bloqué) par un ensemble de nœuds *Z* si et seulement si :

[4] **d** signifie direct

1. *p* contient une séquence i → m → j ou une divergence i ← m → j tel que m ∈ *Z*, ou

2. *p* contient une convergence i → m← j telle que m ∉ Z et tel qu'aucun descendant de m n'appartient à Z.

Un ensemble *Z* d-sépare X de Y si et seulement si Z bloque chaque chemin partant d'un nœud quelconque de X à un nœud quelconque de Y. Si X et Y ne sont pas d-séparés, ils sont alors d-connectés.

L'idée à la base de la d-séparation est simple quand nous attribuons une signification aux flèches dans le graphe. Dans la séquence i → m → j ou dans la divergence i ← m → j si nous conditionnons m (si nous affectons une valeur à m) alors les variables i et j qui étaient dépendantes conditionnellement à m deviennent indépendantes. Conditionner m bloque le flux d'information allant de i à j, c'est à dire qu'une nouvelle connaissance sur i ne pourra plus influer m puisque ce dernier est maintenant connu, et donc m ne changeant plus, il n'aura plus d'influence sur j. Donc i, à travers m, n'a plus d'influence sur j non plus. Dans le cas d'une convergence i → m ← j, représentant deux causes ayant le même effet, le problème est inverse. Les deux causes sont indépendantes jusqu'à ce que nous connaissions leur effet commun. Elles deviennent alors dépendantes. Nous pouvons maintenant définir le concept d'indépendance conditionnelle.

Définition : Indépendance conditionnelle

Deux ensembles de variables X et Y disjoints sont conditionnellement indépendants si et seulement si il existe un ensemble Z (éventuellement vide) qui *d-sépare* X et Y.

3.5 Équivalence entre deux réseaux bayésiens

Un problème relatif aux réseaux bayésiens concerne la vérification si deux ou plusieurs réseaux bayésiens sont réellement différents l'un des autres.

Définition : Équivalence des modèles graphiques

Deux modèles graphiques sont équivalents s'ils représentent le même ensemble d'indépendances conditionnelles [37].

Dans un graphe dirigé, la même fonction de probabilité conditionnelle peut être représentée avec plusieurs graphes. Afin de définir l'équivalence entre deux réseaux bayésiens, il faut en premier lieu définir la notion de **V-structure**. Soient trois noeuds représentant respectivement trois variables A, B et C d'un réseau bayésien. Ces noeuds forment une structure aux flèches convergentes, si A et C sont reliés par la flèche qui va de A vers C, et si B et C sont reliés par la flèche qui va de B vers C. Une illustration de ces liens est donnée par la figure 4.2. Ce type de structure est appelé **V–structure** étant donné sa forme généralement en V. Deux réseaux bayésiens sont dits équivalents si leurs graphes respectifs ont la même structure non dirigée et les mêmes **V-structures.**

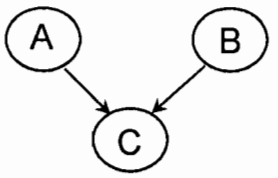


Figure 4.2 : Représentation en V-structure.

4. Mise en place d'un réseau bayésien

La mise en place d'un réseau Bayésien passe par les quatre phases suivantes :

Phase 1 : La préparation des variables

Le premier passage sert à lire les variables et à distinguer les variables discrètes et continues Les variables continues sont découpées en des sous-ensembles ordonnés qui peuvent être travaillés par l'utilisateur. Les variables discontinues sont recensées en termes de fréquence, de façon à permettre un regroupement de certaines occurrences. Les variables discontinues sont alors représentées par autant de noeuds que de valeurs, alors que les variables continues sont modélisées avec la technique des grappes, qui découpe la variable continue en un certain nombre de tranches.

Phase 2 : La sélection des variables

La première lecture étant faite, il faut déterminer les variables d'entrée et les variables de sortie. Les variables de sortie ne peuvent pas être des entrées pour les autres variables. La sélection des variables dans le modèle final s'opère par un classement selon la mesure d'entropie entre les variables.

L'entropie d'un phénomène X s'apprécie par la formule suivante :

$H(X) = -\Sigma P(X) \log P(X)$ avec P(X) qui représente la probabilité d'apparition de X.

Phase 3 : Identification des dépendances

Cela consiste à mesurer la dépendance entre les noeuds puis à les classer par ordre décroissant. La mesure de la dépendance entre deux variables s'effectue en calculant un facteur de dépendance. Ce facteur permet de calculer l'information mutuelle entre deux variables X et Y de la façon suivante :

$$I(X|Y) = (\Sigma\Sigma P(X|Y) \log P(X|Y))/(P(X) \times P(Y)) \quad (4.2)$$

L'information mutuelle entre X et Y représente l'information apportée sur X par la connaissance de Y (et vice versa). Cette expression peut être calculée au moyen des facteurs d'entropie par l'expression suivante :

$$I(X|Y) = H(X) - H(X|Y) \quad (4.3)$$

L'information mutuelle exprime la réduction d'incertitude de l'événement X connaissant Y. La réduction d'incertitude est nulle si les phénomènes X et Y sont indépendants. La connaissance de Y ne nous apporte aucune information sur X. Cette mesure est positive lorsque les distributions de X et Y sont différentes, et est égale à zéro lorsqu'elles sont identiques. La sélection des variables s'opère en fixant un seuil qui permet d'éliminer toutes les variables inférieures à ce seuil.

Phase 4 : La matrice des probabilités

Cette dernière étape consiste à construire les probabilités par un comptage des occurrences entre les différents noeuds.

5. Élagage d'un réseau bayésien

Une première difficulté rencontrée lors de la mise en place d'un réseau bayésien concerne le traitement des variables continues. En effet, même si l'intervalle de valeurs de ces variables sera divisé en tranches, le nombre de ces dernières reste un facteur influent dans la complexité du réseau. Un deuxième facteur de complexité est lié au nombre de connexions entre les noeuds du réseau.

Etant donné que plus le réseau est complexe et plus le temps de calcul augmente et l'interprétation devient plus difficile, la limitation du nombre de variables (ou d'une manière

équivalente, de nœuds) et de liens serait d'un grand secours pour maintenir le réseau utilisable.

- *Le regroupement des valeurs :* La limitation des valeurs peut passer par un regroupement au sein d'une même variable soit sous le contrôle des décideurs, soit par un algorithme de regroupement semblable à la technique de grappe des arbres de décision [38, 39].

- *La limitation des liens :* La limitation des liens se construit en fixant le nombre maximal de « liens parents » pour un noeud. Cette solution présente l'avantage de réduire le temps de calcul, mais peut conduire à la perte de dépendance entre variables. En autorisant un nombre important de parents, il est plus facile de représenter les dépendances complexes qui existent. A l'inverse, un réseau trop pauvre s'avère inapte à la représentation du problème. La recherche d'un réseau optimal est donc un équilibre entre deux extrêmes : une couverture minimale pour assurer une représentation correcte du problème, et une complexité limitée pour maintenir des temps de calcul raisonnables.

6. Applications des réseaux bayésiens

Toute application mettant en oeuvre des connaissances peut donc relever de l'utilisation des réseaux bayésiens, qu'il s'agisse de formaliser la connaissance d'experts, d'extraire la connaissance contenue dans des bases de données, ou d'utiliser le plus rationnellement possible l'un ou l'autre type de connaissance. La plupart des applications qui relèvent des réseaux bayésiens sont des applications d'aide à la décision. Par nature toutes ces applications intègrent un certain degré d'incertitude, qui est très bien pris en compte par le formalisme probabiliste des réseaux bayésiens. A cet effet, des applications existent dans le data-mining [38, 39]. Nous utilisons une base de données pour mettre au point un modèle prédictif. Par définition, une prévision comporte une part d'incertitude. Or la décision, elle doit souvent être binaire [40].

Parmi les applications des réseaux bayésiens, nous pouvons citer :

- L'élaboration des diagnostics (médicaux, de maintenance) où les réseaux bayésiens s'avèrent des outils privilégiés pour tels problèmes. Dans ce cadre, l'application des réseaux bayésiens pour le diagnostic médical a été traitée dans [41]. Dans le domaine médical, les systèmes de décision basés sur le modèle des réseaux bayésiens présentent trois avantages par rapport aux systèmes expert traditionnels : puissance normative, facilité de représentation et d'utilisation de l'incertitude, clarté dans l'acquisition des connaissances [42].

- La sûreté de fonctionnement et l'aide à l'optimisation de la maintenance où l'utilisation des réseaux bayésiens est très bénéfique. En effet, ces modèles permettent de réaliser des analyses de sensibilité ainsi qu'une analyse de données utilisées. Ils sont également utilisés comme aide au diagnostic et aide à la décision. Enfin, l'intégration de multitudes de variables comme les coûts, les experts, ou les tâches de maintenance se fait d'une manière simple en ajoutant des noeuds au réseau bayésien [43].

Notons que pour des applications pratiques, la qualité des algorithmes d'inférence dans les réseaux bayésiens est d'une grande importance. A cet effet, une étude critique des performances de ces algorithmes se trouve dans [44].

7. Avantages et inconvénients des réseaux bayésiens

7.1 Avantages

Grâce à la sémantique globale des réseaux bayésiens et à la manipulation des probabilités conditionnelles, les réseaux bayésiens permettent de combiner des données empiriques et des jugements des décideurs, de représenter explicitement des connaissances incertaines de phénomènes complexes. En effet, ils recherchent le meilleur graphe de connexion entre les variables. Ils apportent donc à l'utilisateur final une connaissance des variables pertinentes et les liens qui unissent ces variables. Ils présentent donc des avantages indéniables au niveau de la lisibilité des relations et au niveau de la prise en compte des effets d'interaction.

De plus et contrairement aux méthodes classiques où l'apprentissage sur des données bruitées est généralement déconseillé car le bruit se propage à tous les niveaux lorsqu'il n'est pas filtré, le réseau bayésien est particulièrement utile pour faire face à des données manquantes ou bruitées. En effet, les différentes entrées étant reliées par une fonction de calcul de probabilité, la non connaissance de trois informations sur quatre est suffisante pour déclencher l'activation d'un noeud intermédiaire ou la reconnaissance d'une sortie. Les réseaux bayésiens se révèlent donc performants pour intégrer l'incertitude et les données manquantes.

7.2 Inconvénients

Le problème de recherche du meilleur réseau est une tâche très consommatrice de puissance informatique. En effet, le nombre de combinaisons possibles « variables-arcs » est de nature combinatoire. Les algorithmes existants déterminent un réseau probable. Néanmoins, ils ne garantissent pas qu'il s'agisse du réseau optimal. Ils recherchent la solution optimale en démarrant d'un réseau simple. Ils évaluent les réseaux dérivés de chaque modification, obtenue par ajout d'un noeud ou d'une dépendance.

Ainsi, la recherche du modèle est très consommatrice en puissance de calcul. Ceci conduit à réduire la formalisation du problème, en collaboration avec des experts du domaine. Le recours aux experts est, en effet, souvent nécessaire pour réduire la complexité initiale, ordonner les variables ou identifier les dépendances les plus importantes. La puissance informatique nécessaire pour construire un réseau bayésien est encore relativement incompatible avec les bases de plusieurs giga-octets.

8. Conclusion

Dans ce chapitre, nous avons présenté les modèles graphiques. Ensuite, nous nous sommes intéressé à l'un des modèles graphiques les plus connus à savoir le réseau bayésien. A cet effet, nous avons caractérisé intuitivement et formellement le réseau bayésien. Nous avons aussi décrit les propriétés d'indépendance fonctionnelle et de d-séparation caractérisant les réseaux bayésiens ainsi que la notion d'équivalence au sein de ces réseaux. Une autre partie non moins importante s'est intéressée à la mise en place et à l'élagage d'un réseau bayésien. Nous avons aussi discuté des applications ainsi que des avantages et inconvénients des réseaux bayésiens.

Chapitre 5 : Implémentation et expérimentation

1. Introduction

Notre projet consiste à créer une base de données catégorielle d'une part et d'autre part de créer un logiciel permettant de trouver la typification catégorielle de chaque phrase. Notre motivation ici est de définir une nouvelle approche d'apprentissage dynamique. Le prototype développé pour accomplir ces deux tâches est implémenté en Java. Les sujets abordés dans les chapitres précédents servent de fondements théoriques à l'algorithme général élaboré pour les besoins de notre projet.

Nous commençons donc par présenter cet algorithme, de façon à donner une vision d'ensemble du traitement. Puis, nous élaborons plus en détail sur les points importants nécessaires à une bonne compréhension du projet, soient la préparation de la base de données catégorielle, l'apprentissage à partir d'un corpus typifié, la typification en elle-même, la validation et le renforcement de l'apprentissage. Une section sur les résultats expérimentaux obtenus ainsi que les possibilités d'optimisation clôture le chapitre.

2. Algorithme général

➢ La première étape consiste à créer une base de données catégorielle (BDCateg), soient les types catégoriels, ceci en se basant sur :

1. la base de données lexicale (BDLEX) et
2. un corpus typifié (validé par un expert du domaine).

La BDLEX contient les lexiques (« Jean ») et leur type grammatical (N : Nom)

« Jean » → N (Nom)

Le corpus typifié contient 74 phrases validées par l'expert du domaine et de cet ensemble de phrases seront extraits les mots et leur type catégoriels correspondants.

« Jean mange »

[n : Jean] - [(d (g s n) n) : mange]

Le type catégoriel de chaque mot d'une phrase est obtenu à partir du corpus.

« Jean »→ n

De retour dans BDLEX, le type grammatical de « Jean » est recherché et la base de données indique que ce mot est de type « N ».

Donc « Jean » est de type grammatical « N » et de type catégoriel « n ».

Tous les mots de la base de données lexicale (BDLEX) ayant « N » comme type grammatical sont copiés dans la base de données catégorielle et le type catégoriel « n » leur est assigné.

Un mot donné peut avoir plus d'un type catégoriel dans le corpus. L'algorithme doit être répété pour chaque type catégoriel de telle façon que tous les types catégoriels possibles pour un mot soient inscrits à la conclusion du programme.

- La deuxième étape du projet consiste à créer un logiciel permettant d'identifier la typification catégorielle de chaque phrase.

En se basant sur :

1. la base de données catégorielle (BDCateg)
2. le corpus
3. l'approche Markovienne
4. la théorie des graphes et l'algorithme de Dijkstra

Nous prenons une phrase du corpus, la décomposons en mots, puis pour chaque mot, une recherche est effectuée dans BDCateg afin d'obtenir tous les types catégoriels associés au mot donné de la phrase. Ensuite, une analyse des probabilités nous révèlera la bonne typification qui devra être validée par l'expert du domaine.

Le même algorithme sera effectué pour les phrases ne faisant pas partie du corpus, qui sont saisies dans la zone appropriée du programme. Une fois la bonne typification ressortie et validée par l'expert du domaine, la nouvelle phrase peut être ajoutée au corpus de base. La figure 5.1 décrit les différentes étapes de notre système.

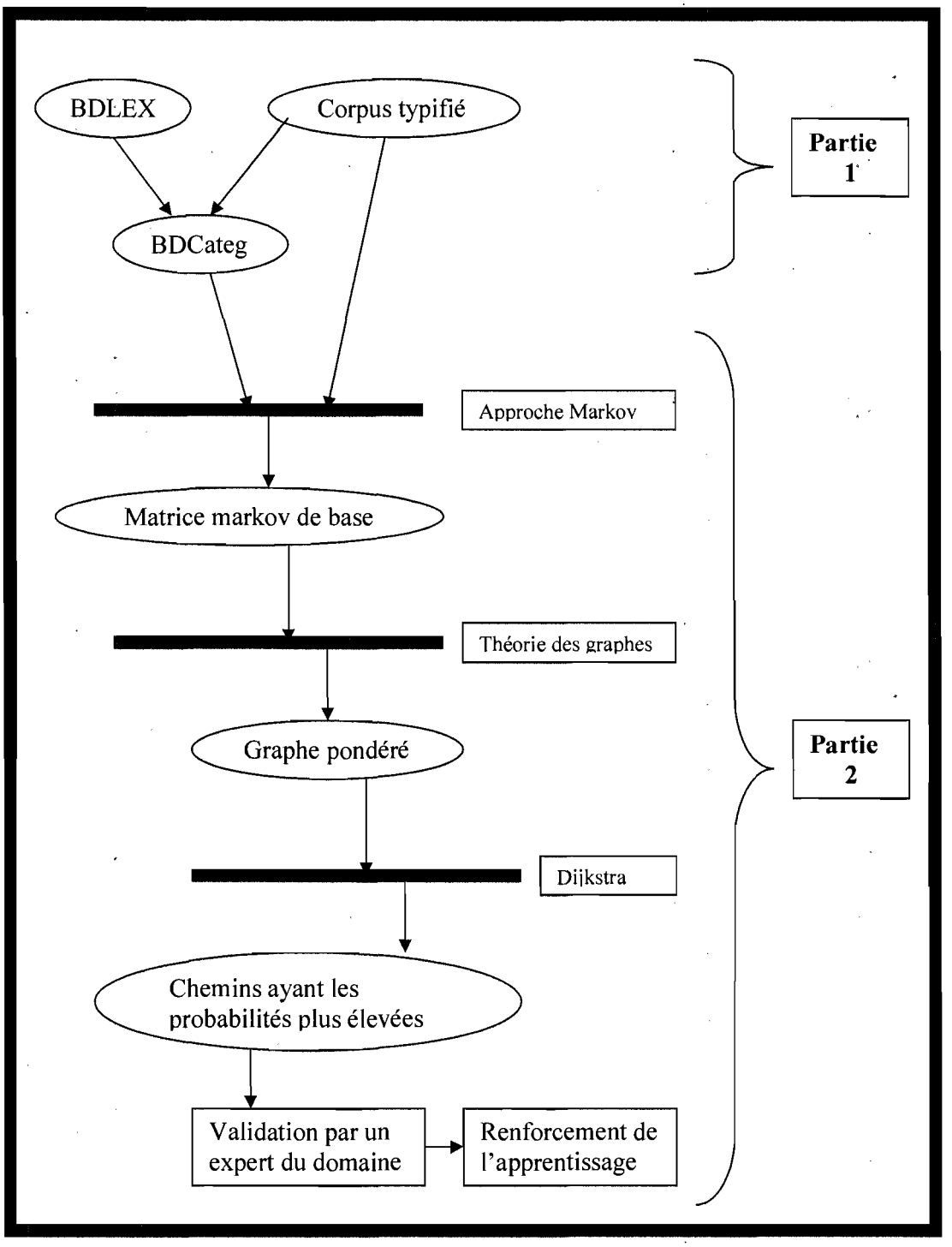


Figure 5.1 : Différentes étapes de notre système

3. Préparation de la base de données catégorielle (BDCateg)

BDLEX est un projet développé dans le cadre du GDR-PRC Communication Homme-Machine, groupe de recherches coordonnées du Ministère de la Recherche et de la

Technologie, et du Centre National de la Recherche Scientifique (France). BDLEX compte 211 270 entrées lexicales de la langue française (écrite et orale) dans la version utilisée.

Chaque entrée de cette base de données est constituée d'un identifiant unique (no_lex), de l'entrée lexicale française elle-même (lexique), de son type grammatical (cg), d'un identifiant de racine (no_racine) et d'un compteur.

La combinaison d'un « lexique » et d'un type grammatical est unique. Cependant, un même lexique peut apparaître plus d'une fois, avec un type grammatical différent. Le compteur a pour utilité de dénombrer les occurrences d'un même lexique dans la base de données. Un échantillon de BDLEX est donné par la figure 5.2.

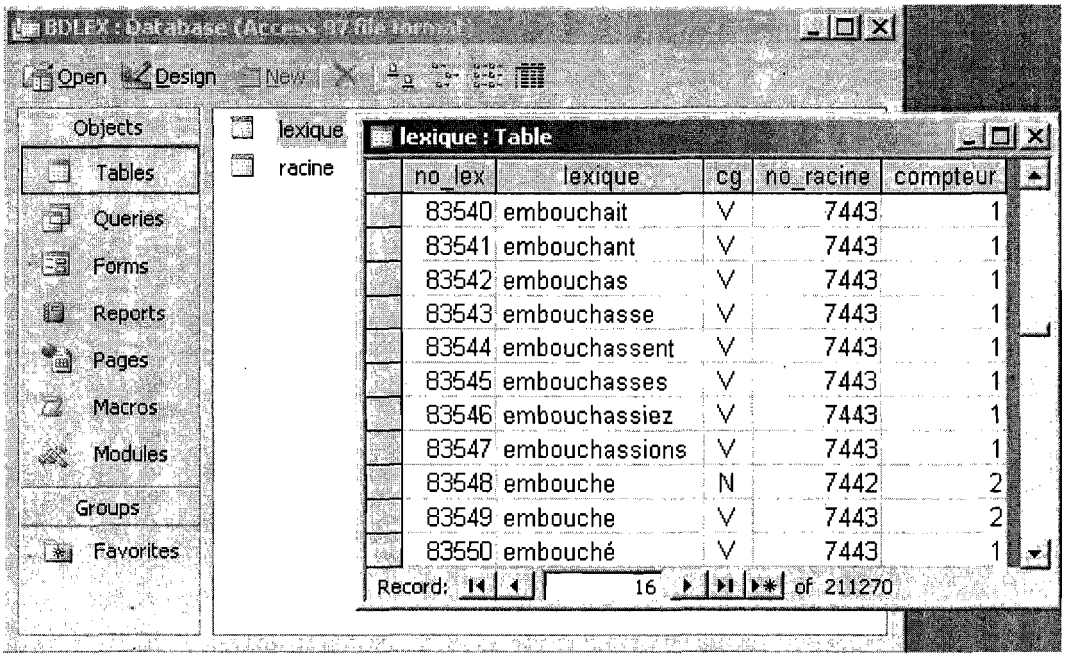


Figure 5.2 : Echantillon de BDLEX.

L'objectif de l'étape de préparation de la base de données catégorielle BDCateg est de passer des entrées lexicales de la langue française (BDLEX) à une base de données de tous les types catégoriels associés aux différents lexiques de la langue (aussi appelé dictionnaire catégoriel).

Les types catégoriels ont la forme X / Y en général (X / Y) / Y ou (X \ Y) / Y.

À partir du corpus, nous prenons par exemple : « Jean mange son repas très rapidement »
[n : Jean] - [(d (g s n) n) : mange] ...

La méthode développée permet de dire la bonne structure de telle sorte qu'elle prend en entrée (d (g s n) n) et retourne le type catégoriel (s \ n) / n. En effet, à « g » correspond l'opérateur « \ » et à « d » correspond le caractère « / ». Ainsi, g s n devient alors s\n. Tandis que (d (g s n) n) et retourne le type catégoriel (s \ n) / n .

Pour passer d'un type grammatical (contenus dans BDLEX) à un type catégoriel (BDCateg a remplir), les phrases du corpus typifié sont décomposés en mots avec leur types catégoriels associés. Ensuite, pour chaque mot, le type grammatical associé est trouvé dans le BDLEX. Pour tous les mots de BDLEX ayant ce type grammatical, nous les copions en associant le type catégoriel initial. Si le mot n'existait pas dans BDLEX alors le mot est ajouté directement dans BDCateg ainsi que son type catégoriel associé (tiré du corpus).

Exemple complet afin d'illustrer la préparation de la base de données des types catégoriels

Soit la phrase suivante commençant par le mot Jean :

Jean .....

[n : Jean] : étant le type catégoriel de Jean que pour cette phrase.

Quant à la base de données grammaticale (*cf.* Table 5.1), elle contient le type grammatical de chaque mot de la langue française.

Numéro du mot	Mot	Type grammatical
.....		
i	Jean	N
.....		

Table 5.1 : Base de données grammaticale.

Le mot numéro i, à savoir « Jean » est de type N (càd, Nom).

Le prototype établit dans un premier lieu une connexion avec la base de données grammaticale et crée une base de données catégorielle (BDCateg) qui est initialement vide. Il parcourt ensuite le corpus et pour chaque mot de ce dernier, il cherche dans la base de données grammaticale (BDLEX), le type grammatical associé. Ensuite, pour tous les mots ayant le même type grammatical dans la BDLEX, il accorde à tous ces mots le type catégoriel (localisé dans le corpus) du mot en traitement.

Exemple :

Soit le mot traité « Jean » de type catégoriel « t » dans la phrase où il a été localisé.

Numéro du mot	Mot	Type grammatical
.....		
K	Jean	N
.....		
k+m	Voiture	N
.....		

Base de données grammaticale

Jean
[t : Jean]

Corpus

Numéro du mot	Mot	Type catégoriel
.....		
S	Jean	t
.....		
s+n	Voiture	t
.....		

Base de données catégorielle

Figure 5.3 : Exemple de génération de la base de donnée catégorielle à partir de la base de données grammaticale et du corpus.

Étant donné que « Voiture » est aussi de type grammatical N (pour Nom), Voiture est ajouté dans la base de donnés catégorielle avec le type catégoriel « t » puisque « Jean » est aussi un nom et a « t » pour type catégoriel, localisé dans le corpus.

Remarque :

Un mot, tel que « Jean », peut avoir plusieurs types catégoriels (suivant sa localisation et son rôle dans les phrases). Cependant, il ne peut avoir qu'un seul type grammatical.

Exemple :

Par exemple, si nous trions la base de données catégorielle par rapport aux mots, nous pouvons trouver la configuration suivante :

Numéro du mot	Mot	Type catégoriel
.....		
i	Jean	n
i+1	Jean	t
i+2	Jean	t/n
.....		

Table 5.2 : Base de données catégorielle.

La figure 5.4 montre un exemple réel de tri par rapport à la base de données catégorielle BDCateg.

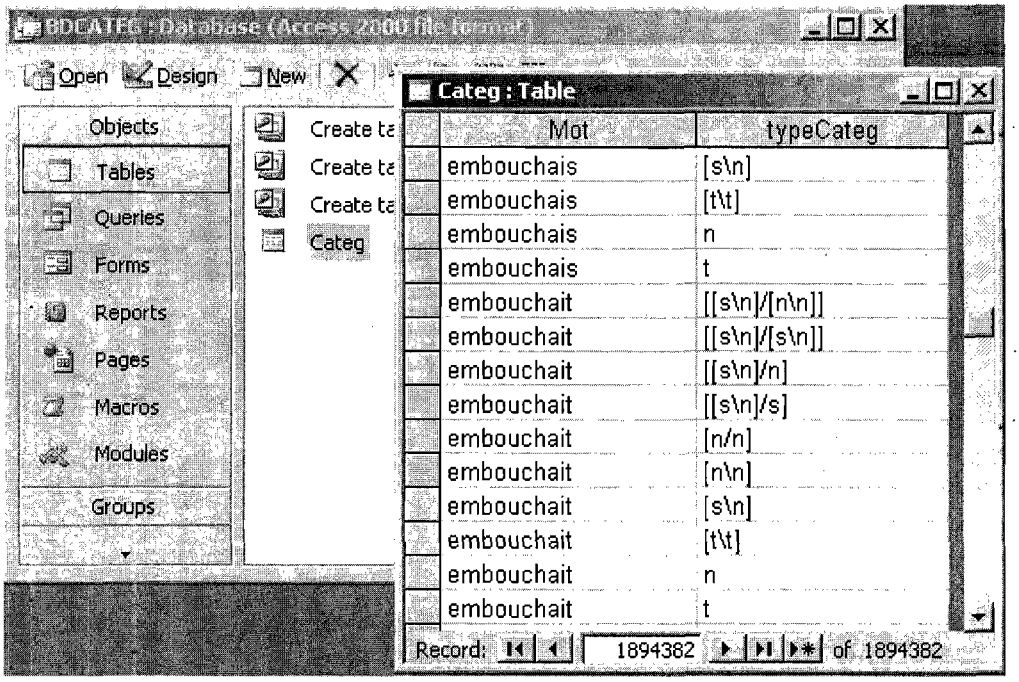


Figure 5.4 : Echantillon de la BDCateg trié par rapport au champs « nom ».

Les catégories syntaxiques de base **n** et **s** sont assignées respectivement aux syntagmes nominaux et aux phrases. Les catégories syntaxiques orientées, construites à partir des types de base au moyen des deux opérateurs de construction de types '/' et '\', sont assignées aux unités linguistiques qui fonctionnent comme des opérateurs. Par exemple, la catégorie **(s\n)/n** est assignée aux verbes transitifs qui sont dès lors considérés comme des opérateurs à deux opérandes, le premier étant l'objet de type **n** positionné à sa droite alors que le second étant le sujet de type **n** positionné à sa gauche.

4. Apprentissage à partir d'un corpus typifié et typification

Le corpus de base pour cette recherche contient 74 phrases. Chaque phrase a été analysée et les différents types catégoriels des mots qui la constituent ont été traités manuellement. Ce constituant représente la partie concernant l'apprentissage. Au fur et à mesure que les phrases sont traitées, les « utilisations » les plus courantes de chaque mot, leurs types catégoriels les plus fréquents, se préciseront. Afin d'obtenir une liste de types catégoriels suffisamment objectifs, nous avons utilisé un corpus de base constitué des 74 phrases ainsi que de leurs typifications catégorielles.

Dans l'étape précédente, nous avons rempli la base de données catégorielle. Dans notre cas, avec le corpus de base utilisé dans nos expérimentations, nous avons 34 types catégoriels. Cette étape permet d'affecter les types catégoriels associés aux termes d'une phrase donnée.

Le prototype commence alors par calculer la matrice de Markov relative à ces types. Il est important de noter que le type « vide », non comptabilisé parmi les 34 types susmentionnés, doit être pris en compte. En effet, chaque phrase est considérée comme étant précédée et suivies par le vide. Par exemple :

vide Jean aime Marie vide

La matrice de Markov correspondante est une matrice carrée dont les colonnes et les lignes reflètent les 34 types, augmentées de deux colonnes qui correspondent au type « vide » qui précède et suit une phrase. Chaque cellule d'indice (j, k) contient une valeur numérique x qui est associé au fait que le type j précède le type k x fois où avec un pourcentage égale à x. En d'autres termes, chaque case reflète le taux de passage d'un type donné à un autre. La table 5.3 montre une matrice de Markov initialisée à nulle.

	vide	type 1	type 2	....	type i	....	type 34	vide
Vide	0	0	0	0	0	0	0	0
type 1	0	0	0	0	0	0	0	0
type 2	0	0	0	0	0	0	0	0
.....	0	0	0	0	0	0	0	0
type i	0	0	0	0	0	0	0	0
.....	0	0	0	0	0	0	0	0
type 34	0	0	0	0	0	0	0	0
Vide	0	0	0	0	0	0	0	0

Table 5.3 : Matrice de Markov initialisée à nulle.

Afin d'illustrer le remplissage de la matrice de Markov, nous allons prendre un exemple pour faciliter la présentation. Soit alors la phrase suivante extraite du corpus de base (càd, des 74 phrases que contient le corpus) avec le type catégoriel de chaque mot l'a constituant :

Jean	court	lentement	Et	Paul	rapidement
n	g s n	(g (g s n) (g s n))	&	n	(g (g s n) (g s n))

Une étape de prétraitement est nécessaire pour mettre sous format adéquat les types catégoriels. En effet, à « g » correspond l'opérateur « \ » et à « d » (non illustré par cet exemple) correspond le caractère « / ». Ainsi, g s n devient alors s\n. Tandis que (g (g s n) (g s n)) devient s\n\s\n.

Le séquencement des types catégoriels est alors comme suit (rappelons que toute phrase est considérée comme étant précédée et suivie par le vide).

Vide	n	s\n	s\n\s\n	&	n	s\n\s\n	vide

Si nous considérons que la matrice de Markov est initialement nulle avant le traitement de cette phrase, le passage du « vide » à « n » est illustré par l'incrémentation d'un pas égal à 1 de la case correspondante. De même pour le passage de « n » à « s\n ». La table 5.4 montre une matrice de Markov remplie au fur et à mesure.

	vide	type 1	n	….	s\n	…..	type i	….	type 34	vide
Vide	0	0	1	0	0	0	0	0	0	0
type 1	0	0	0	0	0	0	0	0	0	0
N	0	0	0	0	1	0	0	0	0	0
…..	0	0	0	0	0	0	0	0	0	0
s\n	0	0	0	0	0	0	0	0	0	0
…..	0	0	0	0	0	0	0	0	0	0
type i	0	0	0	0	0	0	0	0	0	0
…..	0	0	0	0	0	0	0	0	0	0
type 34	0	0	0	0	0	0	0	0	0	0
Vide	0	0	0	0	0	0	0	0	0	0

Table 5.4 : Matrice de Markov remplie au fur et à mesure.

Une fois traitées toutes les séquences de passages contenues dans une phrase, le prototype passe à une nouvelle phrase jusqu'à traiter les 74 phrases contenues dans le corpus. Néanmoins, la matrice obtenue suite à ce traitement ne peut être considérée comme étant une matrice Markov normalisée (*cf.* Table 5.5). En effet, la normalisation consiste à avoir 1 pour somme des valeurs des cases situées dans chaque ligne. Pour cela, le contenu de chaque case est divisé par la somme totale des valeurs des cases qui se trouvent dans la même ligne. Cette somme contient bien sûre la valeur de la case elle-même.

Afin d'expliquer le principe de la normalisation, supposons que nous avons seulement trois cellules dans chaque ligne. Une des lignes est par exemple comme suit :

Cellule 1	Cellule 2	Cellule 3
15	17	19

Table 5.5 : Cellules non normalisées.

La normalisation de la valeur que contient chaque cellule revient à diviser cette valeur par la somme des valeurs de la ligne. Le contenu des cellules devient alors celui montré par Table 5.6.

Cellule 1	Cellule 2	Cellule 3
15/ (15+17+19) = 0,294	17/ (15+17+19) = 0,333	19/ (15+17+19) = 0,373

Table 5.6 : Cellules normalisées.

Une fois le processus de normalisation est appliqué à la matrice, nous obtenons la matrice de Markov telle que dans chaque cellule d'indices (i, j), nous avons la probabilité de passage du type i au type j (càd, la probabilité d'avoir le type i et directement après le type j).

Tel qu'expliqué auparavant, une chaîne de Markov est un système pour une observation donnée. Disons la $k^{ième}$ période la probabilité que le système soit dans un état particulier dépend seulement de son statut à la période k-1.

Clarifions cette définition avec l'exemple suivant :

Supposons (pour simplifier l'explication) que nous avons trois types catégoriels : type1, type2, type3 et que nous avons obtenu les résultats statistiques suivantes

- D'un mot de type1, la probabilité de passer à un mot de type1 est 30 %, à un mot de type 2 30 % et à un mot de type3 40 %
- D'un mot de type 2, la probabilité de passer à un mot de type1 est 40 %, à un mot de type 2 40 % et à un mot de type3 20 %
- D'un mot de type 3, la probabilité de passer à un mot de type1 est 50 %, à un mot de type 2 30 % et à un mot de type3 20 %

Après le passage du mot 1 au mot 2, on passe du mot 2 au mot 3. De cette façon, le type de mot actuel est déterminé seulement par le type du mot précédent.

La matrice donnée par Table 5.7 modélise le problème :

	Type1	Type2	Type3
Type1	0,3	0.4	0.5
Type2	0,3	0.4	0.3
Type3	0,4	0.2	0.2

Table 5.7 : Matrice de transition du système.

Cette matrice est dite de transition du système. L'entrée S_{ji} de la matrice ci-dessus représente la probabilité de transition de l'état correspondant à i jusqu'à l'état correspondant à j.

L'étape d'après consiste à donner la main à l'utilisateur pour qu'il choisisse une phrase à tester. Le choix concerne soit une phrase du corpus de base ou bien une nouvelle phrase.

a. Choix d'une phrase du corpus

Le corpus contient les premières 74 phrases qui ont été déjà validées par l'expert du domaine ainsi que de nouvelles phrases précédemment saisies par l'utilisateur. Le choix concerne donc une de ces phrases.

La première chose réalisée par le prototype concerne la création de la matrice de Markov. En effet, cette dernière doit être générée automatiquement pour prendre en compte les phrases qui ont été ajoutées aux corpus (càd, celles nouvellement saisies par les utilisateurs et dont les résultats respectifs ont été validés par un expert). Ceci est un des buts

de l'apprentissage : Améliorer la qualité de la matrice de Markov au fur et à mesure que de nouvelles phrases sont acceptées par l'expert.

La deuxième étape concerne la recherche des types des mots contenus dans la phrase choisie. Les types respectifs de ces mots sont contenus dans la base de données catégorielle et ont été déterminés lors de la première étape du prototype (*cf.* sous-section 2.1).

Supposons que la phrase soit : Jean court lentement. La table 5.8 associe à chacun des mots « Jean », « court » et « lentement » les types catégoriels qui lui sont associés.

Jean	court	lentement
type 1.1	type 2.1	type 3.1
type 1.2	type 2.2	type 3.2
.....	.....	.....
.....	.....	.....
type 1.n	type 2.m	type 3.k

Table 5.8 : Types catégoriels associés respectivement aux mots « Jean », « court » et « lentement ».

En appliquant la théorie des graphes, nous avons à générer tous les chemins (ou séquences) possibles menant du premier type au dernier. Ainsi, ces chemins sont comme suit :

vide type 1.1 type 2.1 type 3.1 vide
vide type 1.1 type 2.1 type 3.2 vide
.....
vide type 1.1 type 2.1 type 3.k vide

vide type 1.2 type 2.1 type 3.1 vide
vide type 1.2 type 2.1 type 3.2 vide
.....
vide type 1.2 type 2.1 type 3.k vide

.....
.....

vide	type 1.n	type 2.m	type 3.1	vide
vide	type 1.n	type 2.m	type 3.2	vide
		.....		
vide	type 1.n	type 2.m	type 3.k	vide

En utilisant la matrice de Markov, nous calculons la probabilité relative à chaque chemin. Cette probabilité est le résultat du produit de la probabilité de passage d'un type donné à un autre qui le suit immédiatement. Par exemple, soit le chemin suivant :

vide	type 1.1	type 2.1	type 3.1	vide

Dans ce qui suit, la notation P(X|Y) indique le fait que nous sommes arrivés à l'état X sachant que l'état qui le précède directement est Y. Si nous avons,

P(type 1.1 | vide) = 0,13,

P(type 2.1 | type 1.1) = 0,16,

P(type 3.1 | type 2.1) = 0,45,

et

P(vide | type 3.1) = 0,25,

alors la probabilité de chemin est P = P(type 1.1 | vide) * P(type 2.1 | type 1.1) * P(type 3.1 | type 2.1) * P(vide | type 3.1) = 0,13 * 0,16 * 0,45 * 0,25 = 0.00234.

Une fois les probabilités de tous les chemins calculées, le chemin ayant la plus grande probabilité est affiché à l'expert. Ce dernier aura pour rôle de valider ou non la correspondance entre les mots et leurs types catégoriels associés. Si le résultat est considéré comme correct, le processus s'arrête. Sinon, le chemin ayant la probabilité la plus grande parmi le reste des chemins est affiché à l'expert et ainsi de suite jusqu'à la validation de ce dernier. La figure 5.5 montre un écran, de notre application, relatif au choix d'une phrase du corpus.

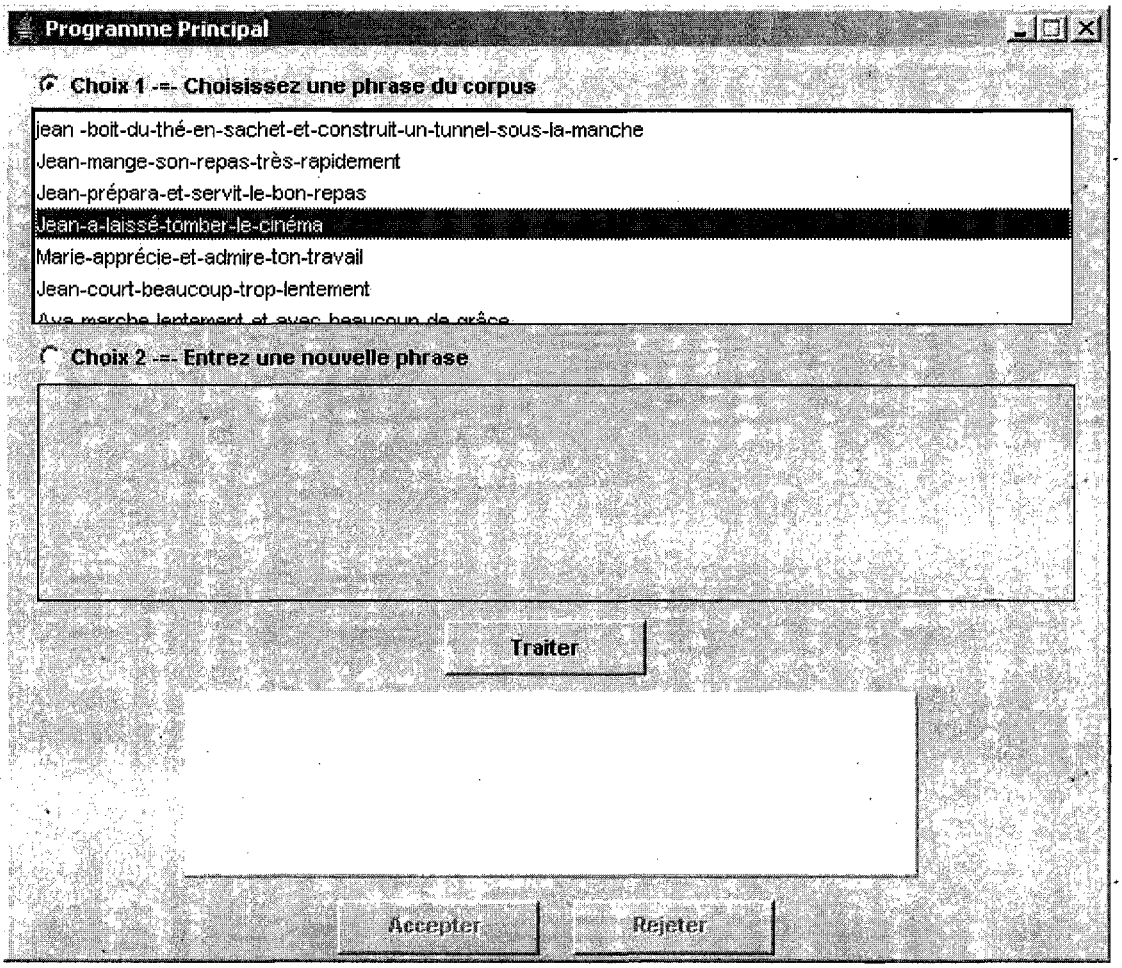


Figure 5.5 : Écran relatif au choix d'une phrase du corpus.

b. Choix d'une nouvelle phrase

Si un des mots de la phrase introduite ne se trouve pas dans la base de données catégorielle, un expert du domaine doit introduire manuellement le type catégoriel de ce mot. Notons que telle opération est très rare vu la taille de la base de données grammaticale et qui contient quasiment tous les mots de la langue française. Elle se limite généralement aux noms propres. Une fois le type catégoriel introduit, le mot en question et son type sont automatiquement ajoutés à la base de données catégorielle.

Par exemple si la phrase est : « Albert joue au football » et le mot Albert ne se trouve pas dans la base de données catégorielle, le type catégoriel du mot « Albert » est à ajouter par l'expert.

Pour les mots déjà existant dans la base de données catégorielle, les différents types qu'ils peuvent prendre seront extraits de la base comme dans le cas du choix d'une phrase du corpus.

Ensuite, la matrice de Markov sera créée en prenant en compte les phrases contenues dans le corpus (les 74 phrases initiales et celles validées par l'expert). Une fois la matrice créée, les différents chemins possibles ainsi que leurs probabilités respectives seront calculés. Les séquences seront alors affichées une par une par valeur de probabilité décroissante jusqu'à validation de l'expert. La figure 5.6 montre un écran relatif au choix d'une phrase du corpus, tiré de notre application.

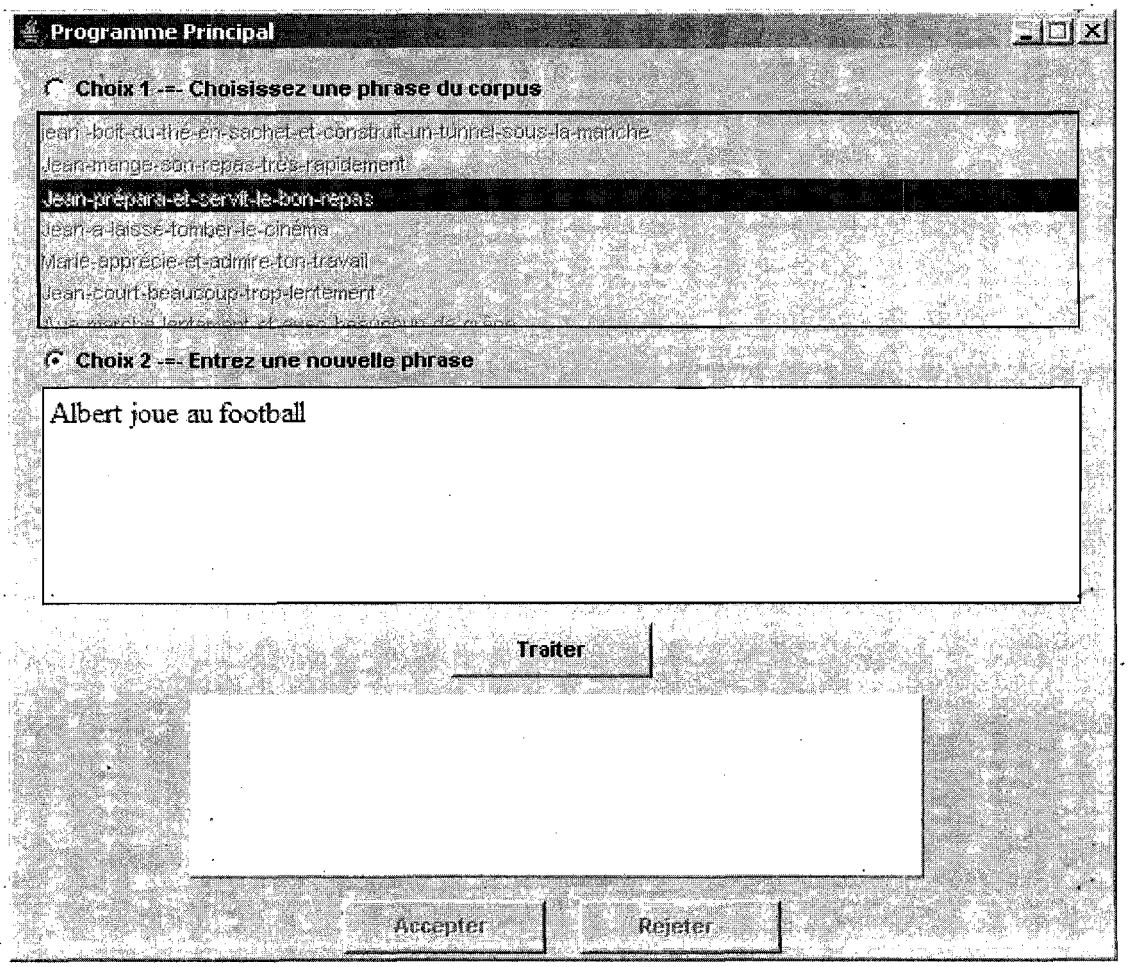


Figure 5.6 : Écran relatif au choix d'une nouvelle phrase

5. Tests et validation

Nous avons testé 58 phrases benchmark choisies de façon à tester différentes formes possibles de phrases. Une partie de ces phrases est déjà existante dans le corpus de base (càd,

font partie des 74 phrases initiales). Le reste est donc formé par de nouvelles phrases. Ce choix est argumenté par le fait que nous avons voulu tester les deux types de choix précédemment décrit. Les résultats de nos tests ont été validés par l'expert du domaine (dans notre cas, M. Biskri).

Exemple de phrases testées et le résultat affiché par notre prototype :

Phrase 1 :

Jean se déplace souvent à pied

La moyenne la plus élevée : 1.6026716352596893E-6 correspond à la séquence

[vide, n, n, [[s\n]/n], [[[s\n]/n]\[[s\n]/n]], n, n, vide]

Est ce que cette séquence est valide? Veuillez entrer votre choix : O pour Oui et N pour Non : O

Phrase 2 :

Ava marche lentement et avec beaucoup de grâce

La moyenne la plus élevée : 7.737141770715585E-6 correspond a la séquence

[vide, n, [s\n], [[s\n]\[s\n]], &, [[[s\n]\[s\n]]/n], [n/n], [n/t], t, vide]

Est ce que cette séquence est valide? Veuillez entrer votre choix : O pour Oui et N pour Non : N

La moyenne la plus élevée suivante : 1.960716672330409E-6 correspond a la séquence

[vide, n, n, [[s\n]\[s\n]], &, [[[s\n]\[s\n]]/n], [n/n], [n/t], t, vide]

Est ce que cette séquence est valide? Veuillez entrer votre choix : O pour Oui et N pour Non : O

Phrase 3 :

L'homme foule allègrement bitument et trottoirs

La moyenne la plus élevée : 6.39264227448832E-6 correspond à la séquence

[vide, [n/t], t, [[s\n]/n], [[[s\n]/n]\[[s\n]/n]], n, &, n, vide]

Est ce que cette séquence est valide? Veuillez entrer votre choix : O pour Oui et N pour Non : N

La moyenne la plus élevée suivante : 9.925582450851352E-7 correspond à la séquence

[vide, [n/t], t, [[s\n]/n], [[[s\n]/n]\[[s\n]/n]], n, *, [n\n], vide]

Est ce que cette séquence est valide? Veuillez entrer votre choix : O pour Oui et N pour Non : N

La moyenne la plus élevée suivante :8.735948445711338E-7 correspond à la séquence

[vide, [n/t], t, [[s\n]/n], [[[s\n]/n]\[[s\n]/n]], n, &, t, vide]

Est ce que cette séquence est valide? Veuillez entrer votre choix : O pour Oui et N pour Non : N

La moyenne la plus élevée suivante :7.839953733330687E-7 correspond à la séquence

[vide, [n/t], t, [[s\n]/n], [[[s\n]/n]\[[s\n]/n]], n, *, t, vide]

Est ce que cette séquence est valide? Veuillez entrer votre choix : O pour Oui et N pour Non : N

La moyenne la plus élevée suivante : 3.374698033289459E-7 correspond à la séquence

[vide, [n/t], t, [[s\n]/n], [[[s\n]/n]\[[s\n]/n]], n, *, n, vide]

Est ce que cette séquence est valide? Veuillez entrer votre choix : O pour Oui et N pour Non : O

Les statistiques relatives à nos tests sont résumées par la table 5.9.

Numéro de la séquence (NbS) validée par l'expert	Nombre de phrases (NbP) validées par la séquence NbS	Pourcentage relatif à chaque groupes de phrases ((NbP/58) * 100)
1	**43**	**74,14%**
2	6	10,34%
3	6	10,34%
4	1	1.72%
5	1	1.72%
9	1	1.72%

Table 5.9 : Statistiques des tests sur 58 phrases benchmark.

Dans la table 5.9, la première colonne présente le numéro de la séquence, dénoté NbS, validées par l'utilisateur comme étant celle adéquate. La deuxième colonne présente le nombre de phrases, dénoté NbP, ayant été validées pour un numéro de séquence NbS donné. La troisième et dernière colonne présente le pourcentage de phrases parmi les 58 testées ayant été validé pour un NbS donné et qui est égal à (NbP/58)*100%. Il est facile de constater que le pourcentage le plus élevé, égal à 74,14%, est celui relatif à la validation de l'expert dès la première séquence. Ceci montre clairement l'efficacité de notre approche étant donné qu'elle présente un degré de précision assez élevé.

6. Renforcement de l'apprentissage

A partir de chaque résultat obtenu, l'expert du domaine doit validé si la séquence correspond bien a la typification appropriée pour la phrase traitée. Si la phrase était déjà incluse dans le corpus alors les probabilités sont ajustées en conséquence, sinon, si c'est une nouvelle phrase, elle est ajoutée au corpus avec la typification spécifiée par l'expert. Dans l'un ou l'autre des cas, ce renforcement de l'apprentissage viendra améliorer l'efficacité du programme pour les utilisations subséquentes.

7. Perspective d'amélioration du prototype

Une amélioration possible du prototype concerne la gestion du nombre d'occurrences d'un mot par rapport aux différents types qui lui sont associés. En effet, nous pouvons comptabiliser le nombre de fois qu'un mot donné prend tel type dans les différentes phrases testées. Après un certain nombre d'apparitions du mot, nous pouvons élaguer les types qui ne sont pas utilisés. Un tel élagage permet d'une part d'alléger la taille de la base de données catégorielle et surtout de réduire le coût du calcul des probabilités des chemins ou ce mot est impliqué.

Exemple :

Soit le mot « Jean » et supposons que nous allons tenter les types non utilisés après 1000 apparitions de ce mot. Si nous supposons que « Jean » ne peut avoir que les types catégoriels suivant : n, t, n\t et t\t et que nous obtenons les statistiques données par la table 5.10, alors nous pouvons élagué les deux lignes relatives aux cas où le type du mot « Jean » est n\t ou t\t.

Mot	Type grammatical	Nombre d'apparitions
Jean	n	690
Jean	t	310
Jean	n\t	**0**
Jean	t\t	**0**

Table 5.10 : Nombre d'apparition des types grammaticaux du mot « Jean ».

8. Conclusion

Dans ce chapitre, nous avons proposé une description de notre prototype. Ce dernier est composé de deux étapes qui sont complémentaires. La première concerne la création d'une base de données catégorielle à partir d'un corpus de base et d'une base de données grammaticale. La seconde étape concerne la détermination dynamique des types grammaticaux des différents mots constituant des phrases choisies par les utilisateurs. A cet effet, le corpus est enrichi au fur et à mesure de l'acceptation de nouvelles phrases par l'expert du domaine. Cet enrichissement permet d'améliorer la qualité et la précision des valeurs contenues dans la matrice de Markov utilisées pour calculer la probabilité de passage d'un type à un autre. Les tests que nous avons effectués sont très encourageants. En effet, dans

plus que 74% des cas, les types grammaticaux des phrases saisies par un utilisateur sont détectés dès la première séquence.

Conclusion générale

Les méthodes de traitement informatique du langage naturel peuvent être réparties suivant deux types d'approches : la première est basée sur des techniques statistiques ou probabilistes appliquées à de grands corpus, tandis que la seconde cherche à formaliser les mécanismes linguistiques à l'aide de traitements symboliques. La première approche est indépendante de la langue mais peut générer des résultats bruités. La seconde approche offre une bien meilleure qualité des résultats mais souffre du fait qu'elle ne prend pas en considération des langues différentes ou des termes spécifiques à des domaines particuliers. Cette approche se base sur la grammaire catégorielle qui, comme décrit au cours de ce mémoire, permet sommairement d'appliquer des types catégoriels aux différents termes d'une phrase donnée en utilisant des règles de production précédemment fixées.

Dans ce mémoire, nous avons présenté une approche hybride permettant de fusionner les mérites des deux approches ce qui permet d'améliorer la qualité du résultat tout en traitant des langues ou domaines différents (moyennant l'adaptation de la base grammaticale à la langue ou domaine choisi). Notre prototype s'exécute en deux étapes. La première permet de générer une base de données catégorielles à partir de la base de données grammaticales. La deuxième étape assure l'affectation des types grammaticaux possibles à un terme donné d'une phrase. Ceci se fait moyennant la construction d'une matrice de Markov prenant en compte l'interaction entre les différents types. Grâce à cette matrice, des chemins d'un graphe formés par des séquences de passage d'un type à un autre peuvent être constitués. La probabilité d'apparition de tels chemins peut être calculée moyennant l'application des fondements relatifs aux réseaux bayésiens. Notre prototype offre à l'utilisateur différentes possibilités d'affectation de types catégoriels aux termes associés. L'utilisateur aura à juger de la pertinence des possibilités offertes. Un des principaux avantages de notre prototype est qu'il est indépendant de la langue ou domaine choisi en choisissant la base grammaticale adéquate.

Bien que les résultats obtenus soient encourageants, plusieurs perspectives de travaux futurs peuvent être envisagées pour l'amélioration de notre travail :

1- L'application de notre prototype à d'autres langues (Arabe, Anglais, etc.) et domaines spécifiques,

2- Comme mentionné dans le chapitre précédent, la réduction de l'ensemble des types grammaticaux qui peuvent être affectés à un mot donné est une piste primordiale à traiter. En effet, ceci permettra de réduire le nombre de chemins à traiter et donc de réduire le coût en traitement de l'analyse de ces derniers. Par ailleurs, la réduction d'un tel ensemble permet de réduire l'espace nécessaire pour le stockage des informations à traiter. En effet, ce point est important car l'espace de stockage peut augmenter « exponentiellement » en fonction du nombre de types.

3- Une autre perspective non moins intéressante concerne la mise en place d'un mécanisme permettant l'analyse de l'interaction entre les mots de façon à améliorer les chances d'affectation du type adéquat à un mot donné.

4- La version actuelle de notre prototype se base sur l'intervention de l'utilisateur pour juger de la validité d'une séquence donnée de types grammaticaux associés aux différents mots d'une phrase. La mise en place d'un outil graphique d'aide à la décision sera utile pour faciliter cette étape importante.

Bibliographie

[1] R. T. Oehrle, E. Bach, D. Wheeler. Categorical grammars and natural language structures. Kluwer Academic Publisher, 1988.

[2] R. Montague. Formal Philosophy: Papers of Richard Montague. New Haven, CT: Yale University Press. Richmond H. Thomason, edition, 1974.

[3] J. Lambek. The Mathematics of Sentence Structure. American Mathematical Monthly, 65, pages 154-170, 1958.

[4] A. Joshi, L. Levy, M. Takahashi. Tree Adjunct Grammars. Journal of the Computer and System Sciences, vol. 10(1), 1975.

[5] I. Biskri. La Grammaire Catégorielle Combinatoire Applicative dans le cadre des Grammaires Applicatives et Cognitives. Ecole des Hautes Etudes en Sciences Sociales, Université de Paris-Sorbonne (Paris IV), France, Juillet 1995.

[6] Y. Bar-Hillel, C. Gaifman, E. Shamir. On categorial and phrase structure grammars, 1960.

[7] J. van Benthem. Language in Action: Categories, Lambdas and Dynamic Logic, volume 130 of Sudies in logic and the foundation of mathematics, North-Holland, Amsterdam, 1991.

[8] M. Moortgat. Categorial type logic. In J. van Benthem and A. ter Meulen, editors. Handbook of Logic and Language. North-Holland Elsevier, chapter 2, pages 93-177, Amsterdam, 1997.

[9] C. Rétoré. Systèmes déductifs et traitement des langues : un panorama des grammaires catégorielles. Rapport de recherche INRIA n ° 3917, avril 2000.

[10] W. Buszkowski, G. Penn. Categorial Grammars Determined from Linguistic Data by Unification, Studia Logica 49 , pages 431-454, December 1990.

[11] M. Kanazawa. Learnable classes of categorial grammars. Studies in Logic, Language and Information. FOLLI, CSLI, Stanford, California, 1998.

[12] I. Biskri, J-P. Desclés. Du Phénotype au Génotype : La grammaire Catégorielle Combinatoire Applicative, Actes de TALN, Marseille, pages 87-96, mai 1996.

[13] M. Steedman. Work in progress: Combinators and grammars in natural language understanding, Summer institute of linguistic, Tucson University, 1989.

[14] B.H. Curry, R. Feys, Combinatory logic , volume 1, North-Holland, 1958.

[15] J-P. Desclés. Langages applicatifs, langues naturelles et cognition, Hermes, Paris, 1990.

[16] N. Haddock. Incremental interpretation and Combinatory Categorial Grammar, Working papers in cognitive science volume 1 : Categorial Grammar, Unification Grammar and Parsing, Edinburgh University, pages 71-84, 1987.

[17] R. Pareschi, M. Steedman. A lazy Way to chart parse with categorial grammars, Acte du colloque ACL, Stanford, 1987.

[18] C. Robert, O. Cogis. La théorie des graphes : Au-delà des ponts de Königsberg : problèmes, théorèmes, algorithmes, Edition Vuibert, Juillet 2003.

[19] M. Sakarovitch. Programmation discrète. Edition Hermann, 1984.

[20] M. Sakarovitch. Graphes et programmation linéaire. Edition Hermann, 1984.

[21] C. Berge. Graphes et Hypergraphes, Edition Dunod, 1973.

[22] W. McCulloch, W. Pitts. A logical Calculus of Ideas Immanent in Nervous Activity, Ibid., 5, pages 115-133, 1943.

[23] M. Gondran, M. Minoux. Graphes et algorithmes, Edition Eyrolles, Janvier 1995

[24] P. Lacomme, C. Prins, M. Sevaux. Algorithmes de graphes, Edition Eyrolles, 2003

[25] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", Proceedings of the IEEE, Vol. 77, No. 2, Février 1989.

[26] G. Kreweras. Graphes, chaînes de markov et quelques applications économiques. Edition Dalloz mathématique, 1972.

[27] B. Ycart, Chaînes de markov, Processus markovien de saut. Cahiers de mathématiques appliquées n° 11-12, Centre de Publication Universitaire, Tunis, 2004.

[28] R. L. Cave, L. P. Neuwirth. Hidden Markov models for English. In Proceedings Symp. Application of Hidden Markov Models to Text and Speech, J. D. Ferguson, Edition, IDA-CRD, Princeton, NJ, USA, pages 16–56, 1980.

[29] L. R. Rabiner, B. H. Juang. An Introduction to Hidden Markov Models, IEEE ASSP magazine, Janvier 1986.

[30] J. Dequier. Chaînes de Markov et applications. Conservatoire National des Arts et Métiers (CNAM), Rhônes-Alpes, Grenoble, France, Juillet 2005.

[31] C. E. Shannon. A mathematical theory of communication. The Bell System technical journal, volume 27, pages 379-423, 1948.

[32] C. E. Shannon. Communications in the presence of noise. In Proceedings of the IRE, volume 37, pages 10-21, 1949.

[33] R. Bellman. Dynamic Programming and Markov Processes. Mathematics of Computation, Vol. 15, No. 74, pages 217-218. April 1961.

[34] F. Jelinek. A Fast Sequential Decoding Algorithm Using a Stack. IBM Journal of Research and Development, volume 13, pages 675-685, 1969.

[35] S. Bouktif. Amélioration de la prédiction de la qualité du logiciel par combinaison et adaptation de modèles, Département d'informatique et de recherche opérationnelle, Faculté des arts et des sciences, Université de Montréal, Mai 2005.

[36] S. Populaire. Introduction aux réseaux bayésiens, Rapport interne du laboratoire SIME. Université de Technologie de Compiègne, 2000.

[37] T. Verma, J. Pearl. Equivalence and Synthesis of Causal Models. Uncertainty in Artificial Intelligence, Amsterdam, 1991.

[38] R. Lefébure, G. Venturi. Le Data mining, Edition Eyrolles, 1999.

[39] P. Adriaans and D. Zantinge. Data mining. Edition Addison-Wesley, Longman, 1997.

[40] A. Becker, P. Naïm. Les réseaux bayésiens. Modèles graphiques de connaissances. Editions Eyrolles, 1999.

[41] P. Leray, O. François. Réseaux bayésiens pour la classification : Méthodologie et illustration dans le cadre du diagnostic médical, Revue d'Intellligence Artificielle 18/2004; pages 169-193.

[42] E. Lepage, M. Fieschi, R. Traineau, J. Gouvernet, C. Chastang. Système d'aide à la décision fondé sur un modèle de réseau bayesien application à la surveillance transfusionnelle Nouvelles Méthodes de Traitement de l'Information en Médecine, Informatique et Santé, volume 5, Paris, Springer-Verlag France, 1992.

[43] F. Corset, G. Celeux, A. Lannoy. Introduction du retour d'expérience dans les réseaux bayésiens, 5ème Congrès International Pluridisciplinaire Qualité et Secret de Fonctionnement, Nancy, France, 18-20 mars 2003

[44] C. S. Kotchi, V. Delcroix, S. Piechowiak. Etude de la performance des algorithmes d'inférence dans les réseaux bayésiens, Journal Electronique d'Intelligence Artificielle, Novembre 2003.

Le modèle de la Grammaire Catégorielle Combinatoire Applicative (GCCA) [4], comme l'ensemble des modèles catégoriels [9] [14] [15] [18], tombe sous un paradigme pour l'analyse des langues qui permet une complète abstraction de la structure grammaticale de sa représentation linéaire ainsi qu'une complète abstraction de la grammaire de son lexique. La GCCA conceptualise les langues comme des systèmes d'agencement d'unités linguistiques (mots, morphèmes, unités lexicales, etc.) dont certaines fonctionnent comme des opérateurs alors que d'autres fonctionnent comme des opérandes. Concrètement, GCCA assigne des catégories syntaxiques à chaque unité linguistique. Les catégories syntaxiques de base N et S sont assignées respectivement aux syntagmes nominaux et aux phrases. Les catégories syntaxiques orientées, construites à partir des types de base au moyen des deux opérateurs de construction de types '/' and '\', sont assignées aux unités linguistiques qui fonctionnent comme des opérateurs. Par exemple, la catégorie (S\N)/N est assignée aux verbes transitifs qui sont dès lors considérés comme des opérateurs à deux opérandes, le premier étant l'objet de type N positionné à sa droite alors que le second étant le sujet de type N positionné à sa gauche (selon la notation de Steedman, X/Y et X\Y sont des types orientés fonctionnels. Une unité linguistique 'u' avec le type X/Y (respectivement X\Y) est considérée comme un opérateur (ou fonction) dont l'opérande est de type Y et positionné à sa droite (respectivement à sa gauche)). Dans notre Article, une unité linguistique 'u' avec un type X sera notée par $'[X : u]'$.

Selon le postulat que la représentation de la langue est sur trois niveaux, la GCCA permet, au moyen de règles, de rendre compte de : (1) la vérification de la bonne formation syntaxique ; (2) la réduction paraphrastique à une forme normale qui est une entrée pour construire l'interprétation ; (3) permettre une analyse fonctionnelle d'un marqueur linguistique (exemple : و *wa* (et), ...).

Les prémisses dans chaque règle sont des concaténations d'unités linguistiques associées à un type orienté qui détermine leur fonctionnement comme opérateurs ou opérandes. La conséquence de chaque règle est une expression applicative typée avec l'introduction d'un combinateur (**B** pour la composition et **C*** pour le changement de type). :

Règles d'Application :

$[X/Y : u_1]$ - $[Y : u_2]$	$[Y : u_1]$ - $[X\backslash Y : u_2]$
----------> ;	<----------
$[X : (u_1\ u_2)]$	$[X : (u_2\ u_1)]$

Règle de changement de type :

[X : u]		[X : u]
----------**>T**	;	**<T**----------
[Y/(Y\X) : (**C*** u)]		[Y\(Y/X) : (**C*** u)]
[X : u]		[X : u]
----------**>Tx**	;	**<Tx**----------
[Y/(Y/X) : (**C*** u)]		[Y\(Y\X) : (**C*** u)]

Règle de Composition fonctionnelle :

$[X/Y : u_1]$-$[Y/Z : u_2]$	$[Y\backslash Z : u_1]$-$[X\backslash Y : u_2]$
----------**>B** ;	**<B**----------
$[X/Z : (\mathbf{B}\ u_1\ u_2)]$	$[X\backslash Z : (\mathbf{B}\ u_2\ u_1)]$
$[X/Y : u_1]$-$[Y\backslash Z : u_2]$	$[Y/Z : u_1]$-$[X\backslash Y : u_2]$
----------**>Bx**;	**<Bx**----------
$[X\backslash Z : (\mathbf{B}\ u_1\ u_2)]$	$[X/Z : (\mathbf{B}\ u_2\ u_1)]$

Illustrons, maintenant, notre modèle par l'analyse d'un énoncé simple en arabe : تدعم الحرية الديمقراطية(que nous transcrirons pour les besoins de l'article au moyen de lettres latines : *Thoudaiimou elhouriathou eddimouqratiyatha*)

1.	[(S/N)/N : *thoudaiimou*]-[N : *elhouriyathou*]-[N : *eddimouqratiyatha*]	
2.	[(S/N)/N: *thoudaiimou*]-[S\(S/N):(**C*** *elhouriyathou*)]-[N: *eddimouqratiyatha*]	**(<T)**
3.	[S/N : (**B** (**C*** elhouriyathou*) thoudaiimou)*]-[N : *eddimouqratiyatha*]	**(<Bx)**
4.	[S : ((**B** (**C*** *elhouriyathou) thoudaiimou) eddimouqratiyatha*)]	**(>)**
5.	((**B** (**C*** *elhouriyathou) thoudaiimou) eddimouqratiyatha*)	
6.	((**C*** *elhouriyathou) (thoudaiimou eddimouqratiyatha*))	**B**
7.	((*thoudaiimou eddimouqratiyatha*) *elhouriyathou*)	**C***
8.	*thoudaiimou eddimouqratiyatha elhouriyathou*	

La langue arabe s'écrit de droite à gauche. Dans la présentation de notre article nous ne tiendrons pas compte de ce facteur car il n'a pas d'influence sur les résultats. Donc nous supposerons que l'arabe se lit de gauche à droite pour faciliter la compréhension des exemples. Le verbe transitif en arabe est un prédicat qui attend ses deux arguments (le sujet et l'objet) à droite. Il est très facile de différencier le sujet de l'objet étant donné la présence dans la phrase d'indices morphologiques. Ainsi concrètement le verbe se voit assigner le type syntaxique catégoriel (S/N)/N. Autrement dit le verbe est considéré comme un opérateur qui prend comme premier opérande l'objet de type N placé à droite de façon à former un opérateur complexe qui prend comme argument le sujet placé à droite et de type N. L'assignation des types est faite à l'étape 1. Les étapes 2 jusqu'à 4 se passent dans le phénotype. Elles consistent à construire à partir du sujet *elhouriathou* un opérateur (**C*** *elhouriathou*) qui à l'étape 3 est composé avec le verbe *thoudaiimou* de façon à produire un opérateur complexe (**B** (**C*** *elhouriathou*) *thoudaiimou*). A l'étape 4 la phrase est syntaxiquement correcte. Un processus déductif naturel de logique combinatoire est effectué dans le génotype à partir de l'étape 5. À l'étape 8 l'interprétation sémantique fonctionnelle est construite. Elle traduit l'ordre effectif opérateur/opérande des unités langagières de l'énoncé initial.

L'exemple traité ici est une phrase dite verbale étant donnée qu'elle commence par le verbe. Il est d'usage en arabe que la plupart des phrases commencent par le verbe, toutefois il n'est pas exclu d'avoir des phrases dites nominales où le sujet se présente avant le verbe :

الحرية تدعم الديمقراطية(*elhouriathou thoudaiimou eddimouqratiyatha*). Et dans ce cas l'analyse se présente comme suit :

1. [N : *elhouriyathou*]-[(S/N)/N : *thoudaiimou*]-[N : *eddimouqratiyatha*]
2. [S/(S/N):(**C*** *elhouriyathou*)]-[(S/N)/N: *thoudaiimou*]-[N: *eddimouqratiyatha*] **(>Tx)**
3. [S/N : (**B** (**C*** elhouriyathou*) thoudaiimou)*]-[N : *eddimouqratiyatha*] **(>B)**
4. [S : ((**B** (**C*** *elhouriyathou) thoudaiimou) eddimouqratiyatha*)] **(>)**

5. ((**B** (**C*** *elhouriyathou) thoudaiimou) eddimouqratiyatha*)
6. ((**C*** *elhouriyathou) (thoudaiimou eddimouqratiyatha*)) **B**
7. ((*thoudaiimou eddimouqratiyatha*) *elhouriyathou*) **C***
8. *thoudaiimou eddimouqratiyatha elhouriyathou*

Le traitement de cet énonce est quasiment le même que le précèdent. Il n'est pas utile donc de le détailler. Ce qui nous importe le plus ici est de montrer la flexibilité et l'adaptabilité de notre modèle. Nous l'avons déjà prouvé pour le français et l'anglais dans d'autres publications.

3. La coordination

En français [10] (ou en anglais), la coordination est une relation implicite ou explicite qui relie des éléments de la même sorte: phrases ou unités linguistiques ayant la même fonction. Elle est principalement représentées par l'utilisation d'un marqueur de coordination comme et (pour le français) ou and (pour l'anglais). Dans la littérature scientifique, il est établi que les éléments coordonnés sont généralement de même natures et fonctions. Cependant, il n'est pas rare de rencontrer des cas de coordination d'éléments de natures différents bien que de fonctions identiques ou certains cas de coordination d'éléments de natures et de fonctions "prétendues" différentes principalement pour l'anglais et le français. Bien que ces distinctions sont significatives pour estimer la couverture d'un modèle, elles ne semblent pas suffire pour les chercheurs qui s'intéressent aussi à une autre distinction : coordination de "constituants" versus coordination de "non-constituants". Ainsi, l'élément [*Paul Sophie*] dans l'énoncé *Jean aime Marie et Paul Sophie* est un non-constituant puisque de façon usuelle un sujet n'est pas suivi de l'objet, mais du verbe. À l'opposé, dans l'énoncé *Les gens préfèrent les femmes cultivées et les hommes courageux*, [*les femmes cultivées*] et *[les hommes courageux]* sont des constituants qui forment des groupes nominaux. Rendre compte de la coordination de non-constituants est devenu un défi significatif. Dans la littérature traditionnelle, que ce soit pour la coordination de constituants ou de non-constituants, la règle générale retenue est : Quand dans les deux membres d'une coordination il y a des éléments identiques la tendance naturelle est de ne pas répéter ces éléments. Une voie explorée par le courant des grammaires génératives et des grammaires transformationnelles, mais vite abandonnée en raison des problèmes théoriques rencontrés [13] [11]. Les modèles syntagmatiques ont paru plus prometteurs avec l'analyse de la coordination avec ellipse au moyen d'une Grammaire HPSG [2] [16], mais ces modèles ne proposent pas des règles assez générales pour l'analyse de l'ensemble des cas de coordination et les solutions proposées apparaissent des fois comme étant ad hoc [12]. C'est d'ailleurs en réaction à cela que le modèle de la TCCG a été proposé pour l'analyse de la coordination en Anglais [1]. TCCG a pour défi, tout en gardant la base grammaticale de HPSG, d'exploiter les performances des Grammaires Catégorielles. Le modèle des Grammaires Catégorielles dans ses multiples versions a montré une capacité de couverture très intéressante, malgré quelques critiques concernant : des aspects de "sur-génération" à

cause principalement des règles de changement de type [5] ou une incapacité à rendre compte de la coordination d'éléments de nature et de fonctions différentes [2] [16].
Tous les modèles catégoriels considèrent que la conjonction de coordination s'applique à deux unités linguistiques ayant la même fonction. L'unité linguistique qui en découle, hérite également de cette fonction. Ce postulat prend forme dans deux règles pour la coordination proposées dans [3], une pour la coordination distributive et une autre pour la non-distributive (nous ne donnerons ici que la règle distributive) :

[X : u_1]-[CONJD : et]-[X : u_2]
----------**<CONJD>**
[X : (**Φ** et u_1 u_2)]

Nous illustrons l'application de la règle de coordination distributive avec l'analyse de l'énoncé en français suivant : *Jean aime Marie et Paul Sophie*

1	[N:*Jean*]-[(S\N)/N:*aime*]-[N:*Marie*]-[CONJD:*et*]-[N:*Paul*]-[N:*Sophie*]	
2	[S/(S\N) : (**C*** *Jean*)]-[(S\N)/N:*aime*]-[N:*Marie*]-[CONJD:*et*]-[N:*Paul*]-[N:*Sophie*]	(**>T**)
3	[S/N : ((**B** (**C*** *Jean*) *aime*)]-[N:*Marie*]-[CONJD:*et*]-[N:*Paul*]-[N:*Sophie*]	(**>B**)
4	[S:((**B** (**C*** *Jean*) *aime*) *Marie*)]-[CONJD:*et*]-[N:*Paul*]-[N:*Sophie*]	(**>**)
5	...-[CONJD:*et*]-[S/(S\N):(**C*** *Paul*)]-[N:*Sophie*]	(**>T**)
6	...-[CONJD:*et*]-[S/(S\N):(**C*** *Paul*)]-[(S\N)\(S/(S\N)):(**C*** *Sophie*)]	(**<T**)
7	...-[CONJD:*et*]-[S\(S/(S\N)):(**B** (**C*** *Paul*)(**C*** *Sophie*))]	(**>Bx**)
8	[S:((**B** (**C*** *Jean*) (**C*** *Marie*)) *aime*)]-[CONJD:*et*]- [S\(S/(S\N)):(**B** (**C*** *Paul*)(**C*** *Sophie*))]	
9	[(S\N)/N:*aime*]-[S\(S/(S\N)):(**B** (**C*** *Jean*) (**C*** *Marie*))]-[CONJD:*et*]- [S\(S/(S\N)):(**B** (**C*** *Paul*)(**C*** *Sophie*))]	
10	[(S\N)/N:*aime*]-[S\(S/(S\N)):(**Φ** *et* (**B**(**C*** *Jean*)(**C*** *Marie*))(**B**(**C*** *Paul*)(**C*** *Sophie*)))]	(**<CONJD>**)
11	[S:((**Φ** *et* (**B** (**C*** *Jean*) (**C*** *Marie*))(**B** (**C*** *Paul*)(**C*** *Sophie*))) *aime*)]	(**<**)
12	((**Φ** *et* (**B** (**C*** *Jean*) (**C*** *Marie*))(**B** (**C*** *Paul*)(**C*** *Sophie*))) *aime*)	
13	(*et* ((**B** (**C*** *Jean*) (**C*** *Marie*)) *aime*) ((**B** (**C*** *Paul*)(**C*** *Sophie*)) *aime*))	(**Φ**)
14	(*et* ((**C*** *Jean*) ((**C*** *Marie*) *aime*)) ((**B** (**C*** *Paul*)(**C*** *Sophie*)) *aime*))	(**B**)
15	(*et* (((**C*** *Marie*) *aime*) *Jean*) ((**B** (**C*** *Paul*)(**C*** *Sophie*)) *aime*))	(**C***)
16	(*et* ((*aime Marie*) *Jean*) ((**B** (**C*** *Paul*)(**C*** *Sophie*)) *aime*))	(**C***)
17	(*et* ((*aime Marie*) *Jean*) ((**C*** *Paul*)((**C*** *Sophie*) *aime*)))	(**B**)
18	(*et* ((*aime Marie*) *Jean*) (((**C*** *Sophie*) *aime*) *Paul*))	(**C***)
19	(*et* ((*aime Marie*) *Jean*) ((*aime Sophie*) *Paul*))	(**C***)

Il est possible, ainsi, d'associer à un énoncé phénotypique *Jean aime Marie et Paul Sophie* une structure applicative génotypique (*et* ((*aime Marie*) *Jean*) ((*aime Sophie*) *Paul*)) dans laquelle il ressort clairement que la conjonction *et* coordonne deux expressions de même type : ((*aime Marie*) *Jean*) et ((*aime Sophie*) *Paul*). Ces deux expressions ne sont que les représentations applicatives des structures phrastiques respectives *Jean aime Marie* et *Paul aime Sophie*. Ce résultat a été rendu possible grâce à la règle de la coordination (étape 10) mais aussi à une procédure de réorganisation des expressions du génotype. Celle-ci est un processus « algorithmisé » qui permet un retour arrière intelligent dans l'arbre syntaxique. Ce processus repose principalement sur l'introduction et l'élimination des combinateurs (étape 8 et 9) [3].

Soient les énoncés en arabe suivants :

(i) *thaâhada ettarafani bi eliîthirafi elmouthabadili wa bi elimthinaii ân ayi thaharoukathin âskariyathin*
تعهد الطرفان بالاعتراف عن وبالامتناع المتبادل عسكرية تحركات أي
(ii) *youkhatibou elibnou abahou bi ihthiramin wa oumahou bi hananin*
يخاطب الإبن أباه باحترام و أمه بحنان
(iii) *yaakoulou elqirdou elmawza wa eddoubou elâssala*
الموز و الدب العسل القرد يأكل
(iv) *asbaha Malikoun mouâliman wa fakhourran bi mansibihi*
أصبح مالك معلما و فخورا بمنصبه

L'énoncé (i) représente la coordination de deux membres de natures différentes mais de fonctions identiques. Les énoncés (ii) et (iii) représentent deux cas différents de coordination avec ellipse. Dans les deux cas le verbe est élidé. Dans les deux cas deux non-constituants sont coordonnés. Dans le premier cas (ii) les non-constituants [*abahou bi ihthiramin*] et [*oumahou bi hananin*] sont formés de l'objet suivi d'un modifieur. Dans le cas de (iii) les non-constituants [*elqirdou elmawza*] et [*eddoubou elâssala*] sont formés du sujet suivi de l'objet. L'énoncé (iv) représente le cas d'une coordination de deux membres de natures et de fonctions différentes. Même si d'un point de vue analytique grammatical classique, [*mouâliman*] et [*fakhourran bi mansibihi*] sont considérés comme « khabar » de *asbaha,* il est quand même clair que les deux membres n'ont pas la même fonction étant

donné que si on inversait leur ordre dans la phrase, celle-ci serait agrammaticale.

Intéressons nous, maintenant, à la présentation de l'analyse catégorielle de l'exemple (ii) :

1. [(S/N)/N : *youkhatibou*]-[N : *elibnou*]-[N : *abahou*]-[(S/N)\(S/N) : *bi-ihthiramin*]-[CONJD : *wa*]-[N : *oumahou*]-[(S/N)\(S/N) : *bi-hananin*]
2. [(S/N)/N : *youkhatibou*]-[S\(S/N):(**C*** *elibnou*)]-[N : *abahou*]-[(S/N)\(S/N) : *bi-ihthiramin*]-[CONJD : *wa*]-[N : *oumahou*]-[(S/N)\(S/N) : *bi-hananin*] **(<T)**
3. [S/N : (**B** (**C*** *elibnou*) *youkhatibou*)]-[N : *abahou*]-[(S/N)\(S/N) : *bi-ihthiramin*]-[CONJD : *wa*]-[N : *oumahou*]-[(S/N)\(S/N) : *bi-hananin*] **(<Bx)**
4. [S : ((**B** (**C*** *elibnou*) *youkhatibou*) *abahou*)]-[(S/N)\(S/N) : *bi-ihthiramin*]-[CONJD : *wa*]-[N : *oumahou*]-[(S/N)\(S/N) : *bi-hananin*] **(>)**
5. [S/(S/N) : (**C*** *elibnou*)]-[S/N : (*youkhatibou abahou*)]-[(S/N)\(S/N) : *bi-ihthiramin*]-[CONJD : *wa*]-[N : *oumahou*]-[(S/N)\(S/N) : *bi-hananin*]
6. [S/(S/N) : (**C*** *elibnou*)]-[S/N : (*bi-ihthiramin* (*youkhatibou abahou*))]-[CONJD : *wa*]-[N : *oumahou*]-[(S/N)\(S/N) : *bi-hananin*] **(<)**
7. [S : ((**C*** *elibnou*) (*bi-ihthiramin* (*youkhatibou abahou*)))]-[CONJD : *wa*]-[N : *oumahou*]-[(S/N)\(S/N) : *bi-hananin*] **(>)**
8. [S : ((**C*** *elibnou*) (*bi-ihthiramin* (*youkhatibou abahou*)))]-[CONJD : *wa*]-[(S/N)\((S/N)/N) : (**C*** *oumahou*)]-[(S/N)\(S/N) : *bi-hananin*] **(<T)**
9. [S : ((**C*** *elibnou*) (*bi-ihthiramin* (*youkhatibou abahou*)))]-[CONJD : *wa*]-[(S/N)\((S/N)/N) : (**B** *bi-hananin* (**C*** *oumahou*))] **(<B)**
10. [S/(S/N) : (**C*** *elibnou*)]-[S/N : ((**B** *bi-ihthiramin* (**C*** *abahou*)) *youkhatibou*)]-[CONJD : *wa*]- [(S/N)\((S/N)/N) : (**B** *bi-hananin* (**C*** *oumahou*))]
11. [S/(S/N) : (**C*** *elibnou*)]-[(S/N)/N : *youkhatibou*]-[(S/N)\((S/N)/N) : (**B** *bi-ihthiramin* (**C*** *abahou*))]-[CONJD : *wa*]- (**B** *bi-hananin* (**C*** *oumahou*))]
12. [S/(S/N) : (**C*** *elibnou*)]-[(S/N)/N : *youkhatibou*]-[(S/N)\((S/N)/N) : (**Φ** *wa* (**B** *bi-ihthiramin* (**C*** *abahou*)) (**B** *bi-hananin* (**C*** *oumahou*)))] **(<CONJD>)**
13. [S/(S/N) : (**C*** *elibnou*)]-[(S/N) : ((**Φ** *wa* (**B** *bi-ihthiramin* (**C*** *abahou*)) (**B** *bi-hananin* (**C*** *oumahou*))) *youkhatibou*)] **(<)**
14. [S : ((**C*** *elibnou*) ((**Φ** *wa* (**B** *bi-ihthiramin* (**C*** *abahou*)) (**B** *bi-hananin* (**C*** *oumahou*))) *youkhatibou*))] **(<)**

15. ((**C*** *elibnou*) ((**Φ** *wa* (**B** *bi-ihthiramin* (**C*** *abahou*)) (**B** *bi-hananin* (**C*** *oumahou*))) *youkhatibou*)))
16. (((**Φ** *wa* (**B** *bi-ihthiramin* (**C*** *abahou*)) (**B** *bi-hananin* (**C*** *oumahou*))) *youkhatibou*))) *elibnou*) **C***
17. ((*wa* ((**B** *bi-ihthiramin* (**C*** *abahou*)) *youkhatibou*) ((**B** *bi-hananin* (**C*** *oumahou*)) *youkhatibou*))) *elibnou*) **Φ**
18. ((*wa* (*bi-ihthiramin* ((**C*** *abahou*) *youkhatibou*)) ((**B** *bi-hananin* (**C*** *oumahou*)) *youkhatibou*))) *elibnou*) **B**
19. ((*wa* (*bi-ihthiramin* (*youkhatibou abahou*)) ((**B** *bi-hananin* (**C*** *oumahou*)) *youkhatibou*))) *elibnou*) **C***
20. ((*wa* (*bi-ihthiramin* (*youkhatibou abahou*)) (*bi-hananin* ((**C*** *oumahou*) *youkhatibou*))) *elibnou*) **B**
21. ((*wa* (*bi-ihthiramin* (*youkhatibou abahou*)) (*bi-hananin* (*youkhatibou oumahou*))) *elibnou*) **C***

Ce qui ressort de l'analyse de ce premier exemple est la similitude dans les étapes de l'analyse d'un énoncé presque semblable en français : *Jean aime* [*Marie tendrement*] *et* [*Sophie sauvagement*] et en anglais : *John loves* [*Mary widly*] *and* [*Petra madly*] qui mettent en jeu la coordination de deux non constituants structurés en un objet suivi d'un modifieur arrière. À quelques règles près, étant donné que l'arabe est une langue principalement VSO, les mêmes grandes étapes se retrouvent particulièrement les phases de réorganisation structurelle que nous retrouvons dans un premier temps à l'étape 5 pour ressortir l'opérande du modifieur arrière *bi-ihthiramin*, l'analyse incrémentale de gauche à droite l'ayant noyé dans la structure temporaire ((**B** (**C*** *elibnou*) *youkhatibou*) *abahou*) et puis dans un second moment aux étapes 10 et 11 pour extraire le premier membre de la coordination. L'expression en 21 est la forme normale qui exprime dans le génotype l'interprétation sémantique fonctionnelle de l'énoncé initiale. Aussi, *wa* est un opérateur qui dans ce cas ci s'identifie au connecteur logique $\wedge$. Par conséquent, de notre énoncé il est possible de déduire un opérateur complexe sous forme conjonctive : (*wa* (*bi-ihthiramin* (*youkhatibou abahou*)) (*bi-hananin* (*youkhatibou oumahou*))) qui s'applique à l'opérande *elibnou*.

L'énoncé (iii) nous intéresse également. Son analyse, avec celle de l'énoncé (ii), permet de démontrer une capacité de couverture importante de la GCCA pour les cas de coordination de non-constituants.

1. [(S/N)/N : *yaakoulou*]-[N : *elqirdou*]-[N : *elmawza*]-[CONJD : *wa*]-[N : *eddoubou*]-[N : *elâssala*]
2. [(S/N)/N : *yaakoulou*]-[S\(S/N) : (**C*** *elqirdou*)]-[N : *elmawza*]-[CONJD : *wa*]-[N : *eddoubou*]-[N : *elâssala*] **(<T)**
3. [S/N : (**B** (**C*** *elqirdou*) *yaakoulou*)]- [N : *elmawza*]-[CONJD : *wa*]-[N : *eddoubou*]-[N : *elâssala*] **(<Bx)**
4. [S : ((**B** (**C*** *elqirdou*) *yaakoulou*) *elmawza*)]-[CONJD : *wa*]-[N : *eddoubou*]-[N : *elâssala*] **(>)**
5. [S : ((**B** (**C*** *elqirdou*) *yaakoulou*) *elmawza*)]-[CONJD : *wa*]-[S/(S/N) : (**C*** *eddoubou*)]-[N : *elâssala*] **(>Tx)**
6. [S : ((**B** (**C*** *elqirdou*) *yaakoulou*) *elmawza*)]-[CONJD : *wa*]-[S/(S/N) : (**C*** *eddoubou*)]-[(S/N)\((S/N)/N) : (**C*** *elâssala*)] **(<Tx)**
7. [S : ((**B** (**C*** *elqirdou*) *yaakoulou*) *elmawza*)]-[CONJD : *wa*]-[S\((S/N)/N) : (**B** (**C*** *eddoubou*) (**C*** *elâssala*))] **(>Bx)**

8. [S : ((**B** (**C*** *elqirdou*) (**C*** *elmawza*)) *yaakoulou*)]-[CONJD : *wa*]-[S\((S/N)/N) : (**B** (**C*** *eddoubou*) (**C*** *elâssala*))]
9. [(S/N)/N : *yaakoulou*]-[S\((S/N)/N) : (**B** (**C*** *elqirdou*) (**C*** *elmawza*))]-[CONJD : *wa*]-[S\((S/N)/N) : (**B** (**C*** *eddoubou*) (**C*** *elâssala*))]
10. [(S/N)/N : *yaakoulou*]-[S\((S/N)/N) : (**Φ** wa (**B** (**C*** *elqirdou*) (**C*** *elmawza*)) (**B** (**C*** *eddoubou*) (**C*** *elâssala*)))] (**<CONJD>**)
11. [S\((S/N)/N) : ((**Φ** wa (**B** (**C*** *elqirdou*) (**C*** *elmawza*)) (**B** (**C*** *eddoubou*) (**C*** *elâssala*))) *yaakoulou*)] (**<**)

12. ((**Φ** wa (**B** (**C*** *elqirdou*) (**C*** *elmawza*)) (**B** (**C*** *eddoubou*) (**C*** *elâssala*))) *yaakoulou*)
13. (wa ((**B** (**C*** *elqirdou*) (**C*** *elmawza*)) *yaakoulou*) ((**B** (**C*** *eddoubou*) (**C*** *elâssala*)) *yaakoulou*)) **Φ**
14. (wa ((**C*** *elqirdou*) ((**C*** *elmawza*) *yaakoulou*)) ((**B** (**C*** *eddoubou*) (**C*** *elâssala*)) *yaakoulou*)) **B**
15. (wa (((**C*** *elmawza*) *yaakoulou*) *elqirdou*) ((**B** (**C*** *eddoubou*) (**C*** *elâssala*)) *yaakoulou*)) **C***
16. (wa ((*yaakoulou elmawza*) *elqirdou*) ((**B** (**C*** *eddoubou*) (**C*** *elâssala*)) *yaakoulou*)) **C***
17. (wa ((*yaakoulou elmawza*) *elqirdou*) ((**C*** *eddoubou*) ((**C*** *elâssala*) *yaakoulou*))) **B**
18. (wa ((*yaakoulou elmawza*) *elqirdou*) (((**C*** *elâssala*) *yaakoulou*) *eddoubou*)) **C***
19. (wa ((*yaakoulou elmawza*) *elqirdou*) ((*yaakoulou elâssala*) *eddoubou*)) **C***

Il est intéressant de remarquer que, dans l'analyse de cet énoncé, la construction du second membre de la coordination passe par l'application aux étapes 5 et 6 respectivement de deux règles de changement de type croisées **(>Tx)** et **(<Tx)**. Ce facteur est en adéquation avec les faits linguistiques propres à la langue arabe qui veulent que l'objet soit précédé dans la structure linéaire de la phrase par le sujet et non par le verbe et que si on admet qu'un « verbe fictif » se place entre l'objet et le sujet (c'est l'interprétation que suppose les changements de type appliqués aux étapes 5 et 6) pour former ce second membre alors le changement de type appliqué à l'objet ne peut être que croisé. Il en va de même pour le changement de type appliqué au sujet. À l'étape 19, l'interprétation de l'énoncé initial est obtenue. *wa*, dans ce cas la aussi, est un opérateur linguistique qui se confond avec le connecteur logique ∧. Une forme conjonctive est, alors déduite : ((*yaakoulou elmawza*) *elqirdou*) ∧ ((*yaakoulou elâssala*) *eddoubou*).

4. Conclusion

D'un point de vue théorique, avec la GCCA, nous montrons, sur des exemples concrets, que ce modèle linguistique fait appel à un formalisme très solide avec un grand pouvoir d'expressivité formelle pour l'analyse de la coordination en arabe et cela grâce à l'utilisation de la logique combinatoire. L'analyse effectuée confirme que les opérateurs catégoriels et les opérateurs de composition — les combinateurs — de la logique combinatoire construisent, à partir de leur structure concaténée, les véritables interprétations des énoncés; ces interprétations sont exprimées dans des termes fonctionnels (opérateur/opérande) ce qui fait ainsi ressortir les membres coordonnés au niveau des expressions du génotype.
Par ailleurs, ces résultats démontrent que malgré les critiques (même si toutes ne sont pas suffisamment vérifiées) concernant les modèles catégoriels en général, il demeure que l'approche catégorielle est assez flexible pour s'adapter à des langues aussi différentes que le français, l'anglais et l'arabe. L'analyse de la coordination pour chacune de ces langues permet d'éprouver la robustesse d'un modèle.

Références

[1] BEAVERS, J., (2004). Type-inheritance Combinatory Categorial Grammar. *In Proceedings of COmputational LINGuistics 2004 (COLING'04)*. Genève, Suisse.

[2] BEAVERS, J., SAG, I.A., (2004). Some Arguments for Coordinate Ellipsis in HPSG. *In Proceedings of the 2004 Head-driven Phrase Structure Grammar Conference (HPSG'04)*. Katholike Universiteit Lueven, Belgique.

[3] BISKRI, I., DESCLÉS, J.P., (2006). Coordination de catégories différentes en français. Dans Faits De Langues. No 28. Editions Ophrys.

[4] BISKRI, I., DESCLÉS, J.P., (1997). Applicative and Combinatory Categorial Grammar (from syntax to functional semantics). *In Recent Advances in Natural Language Processing*, John Benjamins Publishing Company, 71-84.

[5] BRUN, C., (1999). Coordination et analyse du français écrit dans le cadre de la grammaire lexicale fonctionnelle. *In Revue électronique les enjeux de l'information et de la communication.*

[6] CURRY, B.H., FEYS, R. (1958). *Combinatory logic*. North-Holland,Amsterdam.

[7] DESCLÉS, J.P., (1996). Cognitive and Applicative Grammar: an Overview. *In Lenguajes Naturales y Lenguajes Formales*, ed C. Martin Vide, Universitat Rovra i Virgili, pp. 29-60.

[8] DESCLÉS, J. P., (1990). *Langages applicatifs, langues naturelles et cognition*. Hermes, Paris.

[9] DOWTY, D., (2000). The Dual Analysis of Adjuncts/Complements in Categorial Grammar. *In Linguistics.*

[10] GRÉVISSE, M., GOOSE A. (1993). *Le bon usage*. DeBoeck-Duculot, Bruxelles.

[11] HENDRIKS, P., (2003). Coordination. *In P. Strazny (ed.), Encyclopedia of Linguistics, Fitzroy Dearborn,* New York.

[12] KOMAGATA N., (2002). Coordination of Unlike Categories : How not to Distinguish Categories.

[13] MILWARD, D., (1994). Non-Constituent Coordination: Theory and Practice. *In Proceedings of COmputational LINGuistics 1994 (COLING'94).* 935-941.

[14] MORRILL, G., (1994). *Type-Logical Grammar*. Kluwer, Dordrecht.

[15] MOORGAT, M., (1997). Categorial Type Logics, In *Handbook of Logic and Language* , eds. J. Van Benthem and A. Ter Meulen, North Holland, Amsterdam, 93-177.

[16] SAG, I.A., (2003). Coordination and Underspecification. *In Procedings of the 2003 Head-driven Phrase Structure Grammar Conference (HPSG'03).* 267-291.

[17] SHAUMYAN, S. K., (1998). Two Paradigms Of Linguistics: The Semiotic Versus Non-Semiotic Paradigm. *In Web Journal of Formal, Computational and Cognitive Linguistics.*

[18] STEEDMAN, M., (2000). *The Syntactic Process* , MIT Press, Bradford.

Table de matière

Liste des figures

Liste des algorithmes

Liste des tables

Liste des maquettes