TALN 2004, Fès, 19–21 avril 2004

Apprentissage partiel de grammaires catégorielles

Erwan Moreau
LINA - Université de Nantes
2 rue de la Houssinière - BP 92208 - 44322 Nantes cedex 3
Erwan.Moreau@irin.univ-nantes.fr

Résumé - Abstract

Cet article traite de l'apprentissage symbolique de règles syntaxiques dans le modèle de Gold. Kanazawa a montré que certaines classes de grammaires catégorielles sont apprenables dans ce modèle. L'algorithme qu'il propose nécessite une grande quantité d'information en entrée pour être efficace. En changeant la nature des informations en entrée, nous proposons un algorithme d'apprentissage de grammaires catégorielles plus réaliste dans la perspective d'applications au langage naturel.

This article deals with symbolic learning of syntactic rules in Gold's model. Kanazawa showed that some classes of categorial grammars are learnable in this model. But the algorithm needs a high amount of information as input to be efficient. By changing the kind of information taken as input, we propose a learning algorithm for categorial grammars which is more realistic in the perspective of applications to natural language.

Mots-clefs – Keywords

Apprentissage partiel, inférence grammaticale, grammaire catégorielles.
Partial learning, grammatical inference, categorial grammars.

1 Introduction

Malgré la facilité presque surprenante avec laquelle un enfant est capable d'acquérir sa langue maternelle, la réalisation d'un tel processus par la machine est un problème très difficile. Ce problème de l'apprentissage automatique de grammaires consiste à découvrir les règles (syntaxiques) de formation des phrases d'un langage particulier. Plusieurs modèles de formalisation de ce processus existent. Le modèle de Gold (Gold, 1967) est celui que nous utilisons dans cet article, et plus spécifiquement la méthode proposée par Buszkowski (Buszkowski & Penn, 1989) et généralisée par Kanazawa (Kanazawa, 1998) dans ce modèle.

Le modèle d'apprentissage proposé par Gold est très restrictif, c'est pourquoi les premiers résultats obtenus avec ce modèle ont été négatifs. Cependant Kanazawa a montré que certaines classes de langages non triviales sont apprenables, en se servant de l'algorithme proposé par

Buszkoswki pour les grammaires catégorielles. Le mécanisme d'apprentissage proposé est entièrement *symbolique*, c'est-à-dire qu'aucun traitement statistique n'est utilisé : cela implique que le risque d'erreur est réduit à zéro, mais aussi que le bon fonctionnement de l'algorithme d'apprentissage doit satisfaire des contraintes importantes.

L'une de ces contraintes qui font obstacle à l'application de cette méthode au langage naturel concerne la nature des données dont l'algorithme a besoin en entrée : il ne s'agit pas seulement de phrases simples mais de structures particulières, ce qui permet à l'algorithme d'être déterministe et efficace. Ces structures sont une forme "d'arbre de dérivation appauvri" des phrases considérées, dans le formalisme des grammaires catégorielles. La disponibilité de telles structures dans des cas réels (i.e. pas seulement sur des exemples jouets) est loin d'être assurée pour deux raisons : d'une part le formalisme des grammaires catégorielles est très peu utilisé (en grande partie du fait de sa faible expressivité); d'autre part la connaissance de la grammaire sous-jacente est quasiment indispensable à la construction de ces structures, ce qui est un handicap majeur puisque l'objectif est précisément de déduire cette grammaire.

Nous proposons ici un compromis, basé sur la méthode de Kanazawa, qui conserve les avantages de l'apprentissage symbolique tout en éliminant cette contrainte sur les structures. En contrepartie, on considérera qu'une partie de la grammaire est déjà connue, de manière à remplacer l'information apportée par les structures par celle apportée par la *grammaire initiale*. Cette hypothèse est réaliste dans la perspective de l'application aux cas réels, du fait notamment de la lexicalisation totale des grammaires catégorielles. L'inconvénient étant que l'efficacité de l'apprentissage dépend désormais beaucoup de la grammaire initiale.

2 Apprentissage de grammaires catégorielles

2.1 Grammaires AB

Les grammaires catégorielles classiques, nommées aussi grammaires AB, ont été introduites dans (Bar-Hillel *et al.*, 1960). Ces grammaires sont totalement lexicalisées : cela signifie qu'une grammaire est décrite uniquement par son lexique, le lexique étant l'association d'une ou plusieurs catégories à chaque mot du vocabulaire. Les règles utilisées dans les dérivations sont donc *universelles*. Ces règles sont :

$$A/B,\ B \rightarrow A \quad \textit{FA (Forward Application)}$$
$$B,\ B\backslash A \rightarrow A \quad \textit{BA (Backward Application)}$$

Les catégories sont des termes utilisant les opérateurs binaires $/$ et $\backslash$. Intuitivement, une expression est de type A/B (resp. $B\backslash A$) si cette expression est de type A lorsqu'elle est suivie (resp. précédée) par une expression de type B. Une phrase est correcte s'il est possible d'associer à chaque mot l'une de ses catégories, de telle sorte que les règles universelles permettent de transformer cette séquence de catégories en la catégorie spéciale S.

Exemple : Soit G la grammaire constituée du lexique suivant :

$\{$ *Pierre, Marie, Paul* : SN; *aime, déteste* : $(SN\backslash S)/SN$; *qui* : $(SN\backslash SN)/(SN\backslash S)$ $\}$.

La phrase "Pierre, qui aime Marie, déteste Paul" appartient au langage de cette grammaire, comme le montre la dérivation suivante :

SN, $(SN\backslash SN)/(SN\backslash S)$, $(SN\backslash S)/SN$, SN, $(SN\backslash S)/SN$, $SN \Rightarrow SN$, $(SN\backslash SN)/(SN\backslash S)$, $(SN\backslash S)/SN$, SN, $SN\backslash S \Rightarrow SN$, $(SN\backslash SN)/(SN\backslash S)$, $SN\backslash S$, $SN\backslash S \Rightarrow SN$, $SN\backslash SN$, $SN\backslash S \Rightarrow SN$, $SN\backslash S \Rightarrow S$

Une grammaire AB est *rigide* si chaque mot n'est défini que par une seul catégorie. De même, une grammaire est dite *k-valuée* si chaque mot est défini par au plus k catégories.

On peut décrire une dérivation à l'aide d'un arbre de manière classique, en étiquetant les nœuds par la catégorie du constituant qu'ils représentent. De plus, la forme des règles permet aussi de représenter un arbre de dérivation en étiquetant les nœuds seulement par l'identifiant de la règle utilisée (FA ou BA). Une telle structure dans laquelle les feuilles ne sont étiquetées que par un mot est appellée *FA-structure* (pour *Functor-Argument structure*). Cette représention est unique pour un arbre de dérivation donné (voir figure 1). En revanche une FA-structure donnée peut représenter un nombre infini d'arbres de dérivations : dans la figure 1, on peut par exemple remplacer tous les SN par des $(X_1/X_2)/.../X_n$ et la FA-structure reste identique.

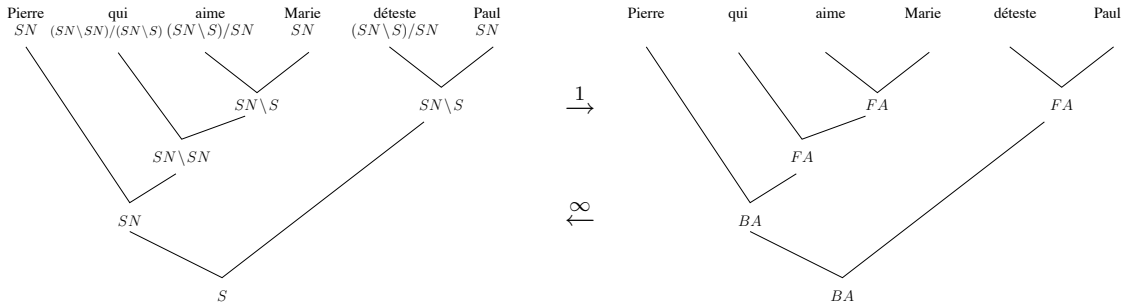

Figure 1: Arbre de dérivation

2.2 L'algorithme RG

L'algorithme RG (pour *Rigid Grammars*), proposé par Buszkowski (Buszkowski & Penn, 1989), apprend la classe des grammaires AB rigides à partir de FA-structures dans le modèle de Gold (Gold, 1967). Dans ce modèle, l'algorithme d'apprentissage doit déduire la grammaire à partir d'une suite infinie de phrases appartenant au langage généré par celle-ci (exemples positifs). Après chaque exemple, l'algorithme effectue une hypothèse, en proposant une grammaire. Si l'algorithme ne change plus d'hypothèse à partir d'une certaine étape, alors celui-ci *converge*. La grammaire-cible est correctement apprise si l'algorithme converge vers cette grammaire (ou une qui lui soit équivalente). Une classe de grammaires est apprenable s'il existe un algorithme qui, pour toute énumération du langage engendré par une grammaire de cette classe, converge vers cette dernière.

2.2.1 Algorithme

L'algorithme RG comporte deux étapes. La première consiste à construire une *grammaire générale* à partir de la séquence de FA-structures fournies comme exemples :

Soit $D = \langle T_1, .., T_n \rangle$ la séquence de FA-structures en entrée.

1. Etiquetage d'une structure T_i :
   (a) la racine de T_i est étiquetée par le type S (le type primitif qui caractérise les phrases correctes).
   (b) en allant de la racine vers les feuilles, les fils de chaque nœud t sont étiquetés de la manière suivante : une nouvelle variable x est créée, avec laquelle le nœud argument est étiqueté. L'autre nœud est étiqueté t/x ou $x \setminus t$ selon qu'il s'agit d'un nœud FA ou BA. Le nœud argument est celui de la branche droite s'il s'agit d'un nœud FA, gauche si c'est un nœud BA.
2. Soit $\langle P_1..P_n \rangle$ l'ensemble des arbres de dérivations construits par étiquetage des FA-structures $\langle T_1..T_n \rangle$. Pour chaque arbre P_i et chaque feuille de cet arbre, une règle $w \mapsto t$ est créée, où w est le mot correspondant à cette feuille et t le type obtenu après étiquetage pour cette feuille. La grammaire ainsi obtenue est la grammaire générale $GF(D)$.

La grammaire ainsi construite génère bien l'ensemble des FA-structures donné en entrée, et donc aussi les phrases décrites par ces structures. Cependant il est évident que cette seule étape ne suffit pas à apprendre le langage décrit, puisque la taille de la grammaire va augmenter indéfiniment, avec chaque nouvel exemple fourni à l'algorithme. C'est la raison pour laquelle la seconde étape *d'unification* doit transformer $GF(D)$ en une grammaire rigide :

1. Pour chaque mot w défini dans $GF(D)$, soit $\mathcal{A}_w$ l'ensemble des types associés au mot w. Soit $\mathcal{A}$ l'union de tous les $\mathcal{A}_w$. Soit σ_u *l'unifieur le plus général* (MGU) de $\mathcal{A}$: un unifieur de $\mathcal{A}$ est une substitution σ telle que pour tout couple de types (t_1, t_2) de tout ensemble $\mathcal{A}_w$ on a $\sigma(t_1) = \sigma(t_2)$. Un unifieur σ_u est le plus général si pour tout autre unifieur σ il existe une substition τ qui permet de passer de σ_u à σ, i.e. $\sigma = \tau \circ \sigma_u$ (voir par exemple (Knight, 1989)).
2. On définit la grammaire $RG(D)$ par $RG(D) = \sigma_u[GF(D)]$: toute règle $w \mapsto t$ de $GF(D)$ est remplacée par $w \mapsto \sigma_u(t)$ dans $RG(D)$. Comme tous les types d'un même mot ont été unifiés, la grammaire obtenue est rigide.

Cet algorithme a quelques qualités intéressantes : il est tout d'abord efficace, puisque sa complexité (en fonction de la taille des exemples) est seulement quadratique. Il peut être utilisé de manière incrémentale, ainsi il n'est pas nécessaire de recalculer la grammaire à partir de l'ensemble des phrases à chaque nouvel exemple. Enfin il produit une unique grammaire solution (la plus générale) quel que soit l'ensemble d'exemples proposés.

2.2.2 Exemple

On considère l'ensemble D de FA-structures représentées (après étiquetage des nœuds) sur la figure 2. La phase d'étiquetage permet d'obtenir la grammaire générale $GF(D)$ suivante :

$$GF(D) = \begin{cases} \text{Pierre} & \mapsto & X_1, X_4, X_9 \\ \text{Marie} & \mapsto & X_3, X_8 \\ \text{Paul} & \mapsto & X_2, X_6 \\ \text{aime} & \mapsto & (X_3 \setminus S)/X_4, X_7/X_8 \\ \text{déteste} & \mapsto & (X_1 \setminus S)/X_2, (X_5 \setminus S)/X_9 \\ \text{qui} & \mapsto & (X_6 \setminus X_5)/X_7 \end{cases}$$

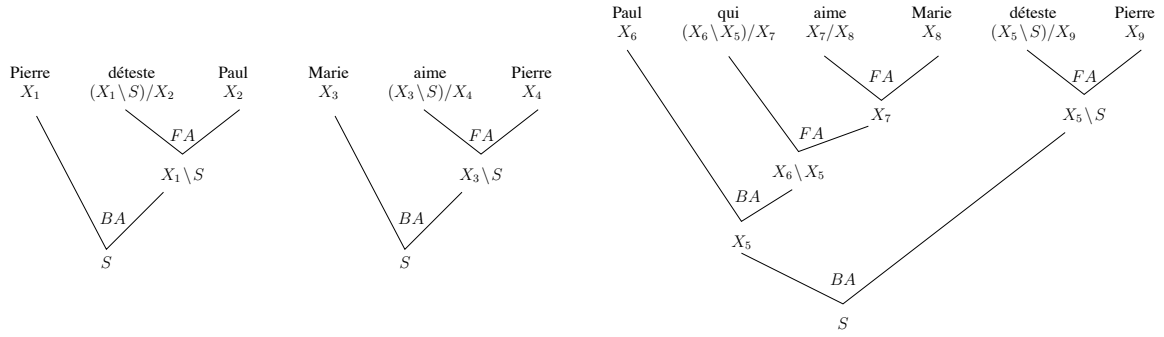

Figure 2: FA-structures après étiquetage

Le calcul du MGU unifie les types suivants : $X_1 = X_4 = X_9$, $X_3 = X_8$, $X_2 = X_6$, $X_7 = (X_3 \setminus S)$, $X_4 = X_8$, $X_1 = X_5$, $X_2 = X_9$, ce qui donne : $X_1 = X_2 = X_3 = X_4 = X_5 = X_6 = X_8 = X_9$ et $X_7 = X_1 \setminus S$. Donc la grammaire $RG(D)$ est :

$$RG(D) = \begin{cases} \text{Pierre} & \mapsto & X_1 \\ \text{Marie} & \mapsto & X_1 \\ \text{Paul} & \mapsto & X_1 \\ \text{aime} & \mapsto & (X_1 \setminus S)/X_1 \\ \text{déteste} & \mapsto & (X_1 \setminus S)/X_1 \\ \text{qui} & \mapsto & (X_1 \setminus X_1)/(X_1 \setminus S) \end{cases}$$

2.3 Apprentissage à partir de phrases plates

L'algorithme RG présente peu d'intérêt, d'une part à cause de la nature des informations qu'il nécessite, et d'autre part à cause de la faible expressivité des grammaires rigides. Mais Kanazawa a aussi montré que la classe des grammaires AB k-valuées est apprenable (au sens de Gold) à partir de "phrases plates" (*flat strings*, i.e. phrases sans FA-structure) (Kanazawa, 1998). L'algorithme qu'il propose consiste en une réduction au cas de l'apprentissage de grammaires rigides à partir de FA-structures. En effet, pour une phrase donnée de longueur n, le nombre de FA-structures possibles est borné (mais grand !), puisqu'il s'agit d'arbres binaires de $n \quad 1$ nœuds, chaque nœud étant étiqueté soit par FA soit par BA. De manière similaire, le nombre de grammaires k-valuées possibles est borné pour une grammaire $GF(D)$ donnée (on considère tous les unifieurs *k-partiels* au lieu de l'unique MGU).

Dans ce cas, l'ensemble des solutions est calculable mais la complexité de l'algorithme devient exponentielle. Costa-Florêncio montre que le problème qui consiste à apprendre la classe des grammaires k-valuées (pour $k > 1$) à partir de FA-structures ainsi que celui qui consiste à apprendre la classe des grammaires rigides à partir de phrases plates sont des problèmes NP-durs (Costa Florêncio, 2001), (Costa Florêncio, 2002). Nicolas a implémenté et testé l'algorithme de Kanazawa pour ce cas, et obtient par exemple 126775 grammaires solutions en fixant seulement la valeur 2 à k, pour de petits exemples (Nicolas, 1999).

3 Apprentissage partiel de grammaires rigides

L'application des algorithmes d'apprentissage de grammaires catégorielles rigides se heurte donc au problème classique du rapport entre quantité d'information en entrée et efficacité de l'algorithme : soit l'algorithme est polynômial mais nécessite des FA-structures trop complexes, soit l'algorithme n'utilise que des phrases plates mais est alors exponentiel, donc tout aussi peu utilisable dans des applications réelles.

3.1 Méthode

Nous proposons ici un compromis dans lequel les informations que constituent les FA-structures sont remplacées par celles fournies par une *grammaire initiale*, de manière à maintenir un niveau acceptable d'efficacité. Comme les grammaires catégorielles sont totalement lexicalisées, l'hypothèse qu'une partie de la grammaire à apprendre soit connue au départ est réaliste : en effet, cette partie est simplement constituée d'un ensemble de mots auxquels sont associés un ensemble de types.

Le programme Prolog ci-contre est à la fois un parser de grammaires AB, un algorithme d'apprentissage de grammaires rigides et un algorithme d'apprentissage partiel. Cet algorithme naïf, qui repose entièrement sur le moteur Prolog pour la recherche d'une dérivation et/ou le calcul des types inconnus du lexique, illustre bien l'importance de l'unification dans le processus de dérivation des grammaires catégorielles.
On peut noter que cet algorithme termine toujours, car le nombre de parenthésages d'une phrase en constituants (non vides) est borné. Cependant il est très peu efficace, surtout dans le cas où un grand nombre de mots sont définis dans le lexique : tous les parenthésages sont testés jusqu'à ce que celui qui correspond à la forme des types soit trouvé.

```
:-op(400,xfx,/).
:-op(400,xfx,\).

regle(A/B,B,A).   % Forward Application (FA)
regle(B,B\A,A).   % Backward Application (BA)

couper_liste(Part1, Part2, Liste) :-
        append(Part1, Part2, Liste),
        Part1 \= [],
        Part2 \= [].

deriv(Lexique, [Mot], T) :-
        member(def(Mot,T), Lexique).
deriv(Lexique, Constituant, T) :-
        couper_liste(C1, C2, Constituant),
        deriv(Lexique, C1, T1),
        deriv(Lexique, C2, T2),
        regle(T1,T2,T).

exemple(X) :-    % 'Marie' de type inconnu :
  Lex = [        %  renvoie X = sn
        def('Pierre', sn),
        def('aime', (sn\s)/sn),
        def('Marie', X)
        ],
  deriv(Lex, [ 'Pierre', 'aime', 'Marie' ], s),
  deriv(Lex, [ 'Marie', 'aime', 'Pierre' ], s).
```

Afin d'optimiser l'utilisation des informations fournie dans la grammaire initiale, on peut procéder de la même manière que pour l'analyse syntaxique (parsing) : chercher tous les types possibles pour tous les constituants complets (i.e. ne contenant aucun mot inconnu). Selon la proportion de mots inconnus dans la phrase et leur répartition, cela permet de limiter l'explosion combinatoire dûe aux différentes structures possibles. L'algorithme proposé ci-dessous utilise une méthode de parsing incrémental de type CYK, de manière à guider la construction de l'arbre de dérivation par les types fournis dans la grammaire initiale.

3.2 Algorithme

L'algorithme RGPL (Rigid Grammars Partial Learning) prend en entrée une phrase $w_1, .., w_n$ et une grammaire initiale G_0, composée de règles associant un type à un mot, de la forme $w \mapsto t$. Il renvoie l'ensemble des grammaires rigides solutions, c'est-à-dire celles contenant les règles de la grammaire initiale et acceptant cette phrase.

RGPL(G_0, $[w_1, w_2, .., w_n]$)
- $Lex \leftarrow \{(W,T) \mid (W \mapsto T) \in G_0\}$ % Initialisation
- créer une matrice vide $M[1..n, 1..n]$
- pour $i \leftarrow 1$ à n faire
  - si $\exists T$ tel que $(w_i, T) \in Lex$ alors
    - $M[i,i] \leftarrow \{(T, Id)\}$
  - sinon
    - créer une nouvelle variable V
    - $Lex \leftarrow Lex \cup \{(w_i, V)\}$
    - $M[i,i] \leftarrow \{(V, Id)\}$
  - fin si
- fin pour
- pour $i \leftarrow 2$ à n faire % Processus de dérivation/apprentissage partiel
  - pour $j \leftarrow i \quad 1$ à 1 faire
    - pour $k \leftarrow j$ à $i \quad 1$ faire
      - pour chaque $(T_l, \sigma_l) \in M[j,k]$ faire
        - pour chaque $(T_r, \sigma_r) \in M[k+1, i]$ faire
          - si $\exists \sigma_u = mgu(\sigma_l, \sigma_r)$ alors
            - créer deux nouvelles variables A, B
            - si $\exists \sigma_{FA} = mgu(\{\{\sigma_u(T_l), A/B\}, \{\sigma_u(T_r), B\}\})$ alors
              - $M[j,i] \leftarrow M[j,i] \cup \{(\sigma_{FA}(A), \sigma_{FA} \circ \sigma_u \circ \sigma_1\}$
            - fin si
            - créer deux nouvelles variables A', B'
            - si $\exists \sigma_{BA} = mgu(\{\{\sigma_u(T_l), B'\}, \{\sigma_u(T_r), B' \setminus A'\}\})$ alors
              - $M[j,i] \leftarrow M[j,i] \cup \{(\sigma_{BA}(A'), \sigma_{BA} \circ \sigma_u \circ \sigma_1\}$
            - fin si
          - fin si
        - fin pour
      - fin pour
    - fin pour
  - fin pour
- fin pour
- $Res \leftarrow \emptyset$ % Application des substitutions compatibles
- pour chaque $(T, \sigma) \in M[1,n]$ faire
  - si $\exists \tau$ tel que $\tau(T) = S$ alors
    - $Res \leftarrow Res \cup \{(\tau \circ \sigma)(Lex)\}$
  - fin si
- fin pour
- renvoyer Res

Fin RGPL

Remarques: Id désigne la substitution identité, et $(\sigma_u \circ \sigma_1) = (\sigma_u \circ \sigma_2)$ car σ_u est défini comme le MGU de σ_1 et σ_2.

Cet algorithme ne décrit le fonctionnement que pour une phrase : le processus d'apprentissage sur un ensemble de phrases consiste à appliquer RGPL sur chaque phrase puis calculer le MGU (s'il existe) sur chaque ensemble de solutions, comme dans l'algorithme de Kanazawa.

De même que dans l'algorithme d'apprentissage à partir de FA-structures, cet algorithme utilise des termes qui peuvent contenir des variables. Ces variables sont progressivement instanciées selon les contraintes imposées par les règles universelles, en particulier dans le cas où le type est combiné avec un type sans variable. Chaque type "réalisable" par un constituant est accompagné de la substitution qui permet de l'obtenir. Cette substitution porte sur les variables associées aux mots inconnus de ce constituant, afin de vérifier lors de chaque combinaison de types que leurs substitutions sont compatibles (par calcul du MGU).

Il est important de noter que malgré son fonctionnement "de type CYK" la complexité de cet algorithme n'est pas polynômiale en général. En effet, le nombre de couples (T, σ) dans chaque case de la matrice M peut croître de manière exponentielle. C'est notamment le cas lorsque tous les mots sont inconnus (la grammaire initiale est vide) : on se trouve alors dans le cas précédent d'apprentissage à partir de phrases plates, et l'algorithme doit calculer l'ensemble des FA-structures possibles.

3.3 Exemple

Soit G_0 la grammaire initiale définie par le lexique suivant[1]

$\{\ un \mapsto SN/N,\ homme \mapsto N,\ poisson \mapsto N,\ nage \mapsto SN\backslash S,\ vite \mapsto (SN\backslash S)\backslash(SN\backslash S)\ \}$

Soit "$un\ homme\ court$" la phrase donnée comme entrée à l'algorithme, où "$court$" est un mot inconnu.

1. *Initialisation :*
$M[1,1] \leftarrow (SN/N, \emptyset)$
$M[2,2] \leftarrow (N, \emptyset)$
$M[3,3] \leftarrow (x_1, \emptyset)$ et $Lex \leftarrow Lex \cup \{(court, x_1)\}$

2. *Dérivation :*

$i=2, j=1, k=1$:	$M[1,2] \leftarrow (SN, \emptyset)$	(FA)
$i=3, j=2, k=2$:	$M[2,3] \leftarrow (x_2, \{x_1 \mapsto N\backslash x_2\})$	(BA)
$i=3, j=1, k=1$:	$M[1,3] \leftarrow (SN, \{x_1 \mapsto N\backslash N\})$	(FA)
$i=3, j=1, k=1$:	$M[1,3] \leftarrow (x_3, \{x_1 \mapsto N\backslash((SN/N)\backslash x_3)\})$	(BA)
$i=3, j=1, k=2$:	$M[1,3] \leftarrow (x_4, \{x_1 \mapsto SN\backslash x_4)\})$	(BA)

3. *Substitutions compatibles avec S :*

$\tau(SN) = S$?	impossible
$\tau(x_3) = S$?	$\{x_1 \mapsto N\backslash((SN/N)\backslash S)\}$
$\tau(x_4) = S$?	$\{x_1 \mapsto SN\backslash S\}$

Après cet exemple, il y a deux grammaires dans l'ensemble des solutions : l'une définit "$court$" par le type $N\backslash((SN/N)\backslash S)$, l'autre par le type $SN\backslash S$. Si l'exemple "$un\ homme\ court\ vite$" apparaît plus tard dans la séquence d'exemples, la première solution sera éliminée, car il n'existe pas de substitution sur les règles universelles permettant de combiner $N\backslash((SN/N)\backslash S)$ avec le type de l'adverbe $vite$, $(SN\backslash S)\backslash(SN\backslash S)$.

[1] On peut noter dans cette grammaire que le type des noms N est un argument du type du déterminant SN/N, et non l'inverse : il s'agit de la notation classique dans les grammaires catégorielles.

3.4 Discussion et perspectives

La contrainte de rigidité

L'algorithme RGPL présenté ci-dessus apprend uniquement des grammaires rigides. Plus exactement, les mots définis dans la grammaire initiale peuvent avoir plusieurs catégories, mais l'algorithme unifie tous les types d'un même mot inconnu. Il ne serait bien sûr pas difficile de gérer des grammaires k-valuées en calculant toutes les possibilités d'unification, mais cela serait contraire à l'objectif puisque l'algorithme deviendrait rapidement inexploitable. On peut par contre envisager pour pallier ce problème que des classes de mots soit définies dans la grammaire initiale : chaque mot inconnu devrait alors appartenir à l'une des classes prédéfinies, ce qui limite fortement les combinaisons de types. Le regroupemement par classes est fréquemment utilisé (par exemple dans (Brill, 1993)), mais suppose que les classes prédéfinies couvrent l'ensemble des combinaisons syntaxiques suceptibles d'apparaître dans les exemples, et ne causent pas de surgénération.

Formalisme

L'étude réalisée dans cet article porte uniquement sur les grammaires AB, la forme la plus simple de grammaires catégorielles. Ce formalisme est plutôt pauvre en termes de représentation des phénomènes linguistiques, c'est pourquoi il est souhaitable d'étendre la méthode proposée à des cadres plus adaptés au langage naturel. Il existe différents travaux qui tendent à montrer que le type d'apprentissage proposé par Kanazawa est généralisable, notamment à d'autres formes de grammaires catégorielles telles les grammaires de Lambek et minimalistes (Bonato & Retoré, 2001) (même si cela pose d'autres problèmes de complexité, voir (Pentus, 2003)). D'autres formalismes, comme les grammaires de liens (Sleator & Temperley, 1991), sont aussi suceptibles de permettre ce type d'apprentissage.

La grammaire initiale

L'intérêt de la méthode d'apprentissage que nous proposons repose entièrement sur l'existence d'une grammaire initiale. Celle-ci doit être suffisamment complète pour qu'un nombre significatif de mots soient connus dans les phrases de la séquence à apprendre. Dans ce cadre, on peut tirer profit de la "loi de Zipf", utilisée notamment par l'étiqueteur de Brill (Brill, 1993). Celle-ci garantit que, sur l'ensemble des mots d'un texte, une faible proportion des mots suffit à représenter une grande partie du texte (en nombre d'occurences). Or précisément ce sont les mots les plus fréquents d'un langage qui sont le plus facile à répertorier et définir (mots grammaticaux tels que déterminants, pronoms, prépositions, conjonctions, etc.), soit manuellement soit par conversion de dictionnaires existants dans d'autres formalismes.

4 Conclusion

Dans le domaine de l'inférence grammaticale, l'apprenabilité des classes de langages est souvent étudiée indépendamment d'une éventuelle mise en pratique de cet apprentissage. Du strict

point de vue de l'apprenabilité dans le modèle de Gold, aucune distinction n'est nécessaire entre les cas d'apprentissage de grammaires rigides à partir de FA-structures et de grammaires k-valuées à partir de phrases plates, même si le premier est peu utile et le second quasiment irréalisable.

Dans la perspective d'applications "réelles", nous avons proposé une adaptation des algorithmes d'apprentissage de grammaires catégorielles. L'approche étudiée est moins contraignante sur la forme des entrées de l'algorithme, tout en restant relativement réaliste du point de vue de l'efficacité. Néanmoins il reste plusieurs questions à résoudre avant de pouvoir appliquer des algorithmes d'apprentissage symbolique de règles syntaxiques au langage naturel : l'expressivité des classes de langages apprenables ainsi que le formalisme utilisé pour les représenter doivent être améliorés (les pronoms, par exemple, sont difficilement représentables dans les grammaires AB), et il reste à déterminer comment et selon quels critères constituer la grammaire initiale.

Références

BAR-HILLEL Y., GAIFMAN C. & SHAMIR E. (1960). On categorial and phrase structure grammars.

BONATO R. & RETORÉ C. (2001). Learning rigid lambek grammars and minimalist grammars from structured sentences. In L. POPELÍNSKÝ & M. NEPIL, Eds., *Proceedings of the 3d Workshop on Learning Language in Logic*, p. 23–34, Strasbourg, France.

BRILL E. (1993). *A Corpus-Based Approach to Language Learning*. PhD thesis, Computer and Information Science, University of Pennsylvania.

BUSZKOWSKI W. & PENN G. (1989). *Categorial grammars determined from linguistic data by unification.* Rapport interne TR-89-05, Department of Computer Science, University of Chicago.

COSTA FLORÊNCIO C. (2001). Consistent Identification in the Limit of the Class k-valued is NP-hard. In P. DE GROOTE, G. MORRILL & C. RETORÉ, Eds., *Logical Aspects of Computational Linguistics, 4th International Conference, LACL 2001, Le Croisic, France, June 27-29, 2001, Proceedings*, volume 2099 of *Lecture Notes in Computer Science*, p. 125–138: Springer-Verlag.

COSTA FLORÊNCIO C. (2002). Consistent Identification in the Limit of Rigid Grammars from Strings is NP-hard. In P. ADRIAANS, H. FERNAU & M. VAN ZAANEN, Eds., *Grammatical Inference: Algorithms and Applications 6th International Colloquium: ICGI 2002*, volume 2484 of *Lecture Notes in Artificial Intelligence*, p. 49–62: Springer-Verlag.

GOLD E. (1967). Language identification in the limit. *Information and control*, **10**, 447–474.

KANAZAWA M. (1998). *Learnable classes of categorial grammars*. Cambridge University Press.

KNIGHT K. (1989). Unification: A multidisciplinary survey. *ACM Computing Surveys*, **21**(1), 93–124.

NICOLAS J. (1999). *Grammatical inference as unification* . Rapport interne 3632, INRIA. également rapport IRISA PI1265.

PENTUS M. (2003). *Lambek calculus is NP-complete.* Rapport interne TR-2003005, CUNY Ph.D. Program in Computer Science.

SLEATOR D. D. K. & TEMPERLEY D. (1991). *Parsing English with a Link Grammar*. Rapport interne CMU-CS-TR-91-126, Carnegie Mellon University, Pittsburgh, PA.