TALN 2003, Batz-sur-Mer, 11–14 juin 2003

Apprentissage discriminant pour les Grammaires à Substitution d'Arbres

Antoine Rozenknop, Jean-Cédric Chappelier, Martin Rajman
Laboratoire d'Intelligence Artificielle - Ecole Polytechnique Fédérale de Lausanne
CH-1015 Lausanne, Switzerland
{Antoine.Rozenknop,Jean-Cedric.Chappelier,Martin.Rajman}@epfl.ch

Mots-clefs – Keywords

STSG, Gibbs-Markov, Maximum d'Entropie, Vraisemblance Conditionnelle
STSG, Gibbs-Markov,Maximum Entropy, Conditionnal likelihood

Résumé - Abstract

Les grammaires stochastiques standards utilisent des modèles probabilistes de nature générative, fondés sur des probabilités de récriture conditionnées par le symbole récrit. Les expériences montrent qu'elles tendent ainsi par nature à pénaliser les dérivations les plus longues pour une même entrée, ce qui n'est pas forcément un comportement souhaitable, ni en analyse syntaxique, ni en reconnaissance de la parole. Dans cet article, nous proposons une approche probabiliste non-générative du modèle STSG (*grammaire stochastique à substitution d'arbres*), selon laquelle les probabilités sont conditionnées par les feuilles des arbres syntaxiques plutôt que par leur racine, et qui par nature fait appel à un apprentissage discriminant. Plusieurs expériences sur ce modèle sont présentées.

Standard stochastic grammars use generative probabilistic models, focussing on rewriting probabilities conditioned by the rewritten symbol. Such grammars therefore tend to give penalty to longer derivations of the same input, which could be a drawback when they are used for analysis (e.g. speech recognition). In this contribution, we propose a novel non-generative probabilistic model of STSGs (*Stochastic Tree Substitution Grammars*), where probabilities are conditioned by the leaves of the syntactic trees (i.e. the input symbols) rather than by the root. Several experiments of this new model are presented.

1 Motivations

Les grammaires stochastiques standards sont des modèles probabilistes génératifs dans lesquels les probabilités sont conditionnées par le symbole à récrire. Par exemple, les probabilités des grammaires hors-contextes stochastiques (SCFG) sont les probabilités de récrire la partie gauche de la règle connaissant sa partie droite. Les grammaires à substitution d'arbres stochastiques (STSG), utilisées dans le cadre DOP (Data-Oriented Parsing) (Bod, 1998; Chappelier & Rajman, 2001), sont quant à elles des grammaires dont les règles sont des arbres syntaxiques, appelés "*arbres élémentaires*". Ces arbres élémentaires sont combinés à l'aide de l'opérateur de substitution pour donner des dérivations d'arbres d'analyse complets. De plus, à chaque arbre élémentaire τ est associée la probabilité $p(\tau)$ d'utiliser l'arbre élémentaire pour récrire son symbole racine dans une dérivation. Les modèles SCFG et STSG sont équivalents d'un point de vue structurel[1], mais clairement différents d'un point de vue probabiliste. De fait, les STSG peuvent capturer plus de dépendances probabilistes que les SCFG, qui sont restreintes à des règles hors-contextes équivalentes à des arbres élémentaires de profondeur strictement égale à 1.

Utilisées en reconnaissance de la parole ou en analyse syntaxique, ces grammaires présentent différents inconvénients : les mécanismes d'apprentissage standards des SCFG mènent à des modèles affectant des estimations biaisées aux probabilités des arbres syntaxiques (Johnson, 1998) ; pour sa part, le modèle DOP est connu pour sur-estimer les probabilités des arbres de grande profondeur et critiqué pour le manque de justification de sa procédure d'apprentissage[2] (Bonnema *et al.*, 1999).

Le but de cet article est de présenter des modèles log-linéaires discriminants de grammaires à substitution d'arbres stochastiques qui évitent ces inconvénients : les modèles Gibbsiens de Grammaires à Substitution d'arbres (GTSG). Nous en donnons d'abord une définition formelle. Puis nous présentons les éléments-clefs pour l'apprentissage des paramètres de ces modèles à partir d'un corpus, avant de conclure par une comparaison des GTSG aux STSG standards dans une tâche d'analyse syntaxique.

2 Le modèle GTSG

Le point de départ du modèle est de considérer une probabilisation des TSG qui ne se base pas sur la probabilité d'un arbre conditionnellement à sa racine, mais plutôt conditionnellement à ses feuilles, ce qui semble plus naturellement correspondre à une tâche d'analyse syntaxique. M. Collins traite le problème comme un problème de classification en proposant une approche à base de "Voted Perceptron" (Collins & Duffy, 2002).

Nous utilisons ici une approche de type Gibbs-Markov (Lafferty, 1996), décrite dans (Rozenknop, 2002) pour les grammaires hors-contextes, qui permet de choisir les traits pris en compte pour le calcul des probabilités conditionnelles. Dans le cas de l'analyse, le modèle repose sur des probabilités d'analyse **conditionnellement aux phrases**, les traits étant les arbres élémentaires apparaissant dans les dérivations des analyses.

Cette approche implique que nous imposons *a priori* que la probabilité conditionnelle d'une dérivation d connaissant la phrase w suit une distribution de Gibbs :

[1] Le même langage est reconnu et les mêmes arbres sont engendrés pour une même phrase.

[2] i.e. la façon de calculer ses paramètres à partir d'un corpus d'exemples.

$$p(d|w) = \frac{1}{Z_\lambda(w)} e^{\sum_{\tau \in \mathcal{R}} \lambda_\tau f_\tau(d)} = \frac{1}{Z_\lambda(w)} e^{\lambda \cdot f(d)} \quad (1)$$

où $Z_\lambda(w)$ est le facteur de normalisation[3] $\sum_{d \Rightarrow w} e^{\sum_{\tau \in \mathcal{R}} \lambda_\tau f_\tau(d)}$, $f_\tau(d)$ est le nombre d'occurrences de l'arbre élémentaire τ dans la dérivation d, $\mathcal{R}$ est l'ensemble des arbres élémentaires de la grammaire, $\lambda = (\lambda_{\tau_1}, \ldots, \lambda_{\tau_n})$ est le vecteur des paramètres associés aux arbres élémentaires, et $f(d) = (f_{\tau_1}(d), \ldots, f_{\tau_n}(d))$.

De plus, comme pour une STSG standard, la probabilité conditionnelle d'un arbre t connaissant la phrase w est définie comme la somme des probabilités conditionnelles de toutes ses dérivations[4] :

$$p_\lambda(t|w) = \sum_{d \Rightarrow t} p_\lambda(d|w) \quad (2)$$

Nous appelons *Gibbsian Tree Substitution Grammar* (GTSG) une STSG ainsi probabilisée.

Notons que la différence principale entre GTSG et STSG tient dans la probabilisation du modèle et dans l'algorithme d'apprentissage associé. Cependant, des algorithmes identiques peuvent être utilisés pour effectuer une analyse syntaxique, i.e. déterminer l'analyse la plus probable (MPP[5]) d'une phrase.

Exemple jouet

L'exemple suivant illustre les différences entre STSG et GTSG. Considérons une TSG contenant les arbres élémentaires de la figure 1.[6]

Là où la STSG associe des probabilités $p_1, \ldots, p_5$ aux arbres élémentaires, la GTSG utilise des valeurs réelles non contraintes $\lambda_1, \ldots, \lambda_5$, appelées *potentiels* et intégrées dans le modèle probabiliste par l'équation (1).

Supposons alors que nous voulions utiliser ces grammaires pour analyser la phrase "*Regarde la femme avec un chapeau*", qui peut être associée aux analyses (A) et (B) de la figure 2.

Avec la STSG, les probabilités des dérivations sont respectivement :[7]

$$p_{STSG}(d_1(A)) = p_1 \cdot p_3 \cdot p_4^2 \cdot p_5 \qquad \text{et} \qquad p_{STSG}(d_1(B)) = p_2 \cdot p_4^2 \cdot p_5$$

Avec la GTSG, l'équation (2) impose que le potentiel associé à une dérivation soit la somme des potentiels de ses arbres élémentaires, d'où :

$$\lambda(d_1(A)) = \lambda_1 + \lambda_3 + 2\lambda_4 + \lambda_5 \qquad \text{et} \qquad \lambda(d_1(B)) = \lambda_2 + 2\lambda_4 + \lambda_5$$

Les probabilités des dérivations $d_1(A)$ et $d_1(B)$ *conditionnellement aux feuilles* w sont alors :

$$p_{GTSG}(d_1(A)|w) = \frac{e^{\lambda(d_1(A))}}{e^{\lambda(d_1(A))} + e^{\lambda(d_1(B))}} \qquad \text{et} \qquad p_{GTSG}(d_1(B)|w) = \frac{e^{\lambda(d_1(B))}}{e^{\lambda(d_1(A))} + e^{\lambda(d_1(B))}}$$

[3] $d \Rightarrow w$ représente l'ensemble des dérivations menant à w.

[4] Avec une STSG, un arbre d'analyse peut avoir plusieurs dérivations différentes, contrairement au cas des CFG.

[5] MPP = Most Probable Parse.

[6] Notez que ces arbres sont de profondeur 1, et que la grammaire peut alors aussi être considérée comme une CFG ; ce n'est pas le cas en général, mais cela simplifie ici notre illustration.

[7] Avec cet exemple élémentaire, chaque analyse possède une seule dérivation.

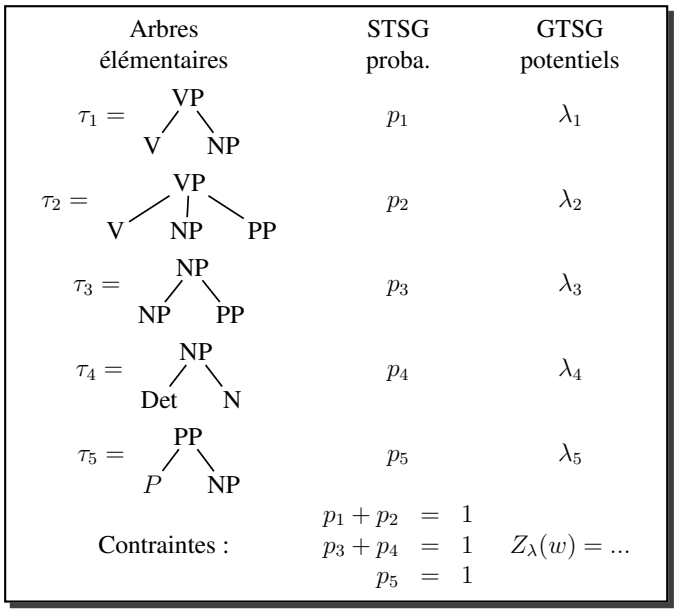

FIG. 1 – Exemple STSG/GTSG.

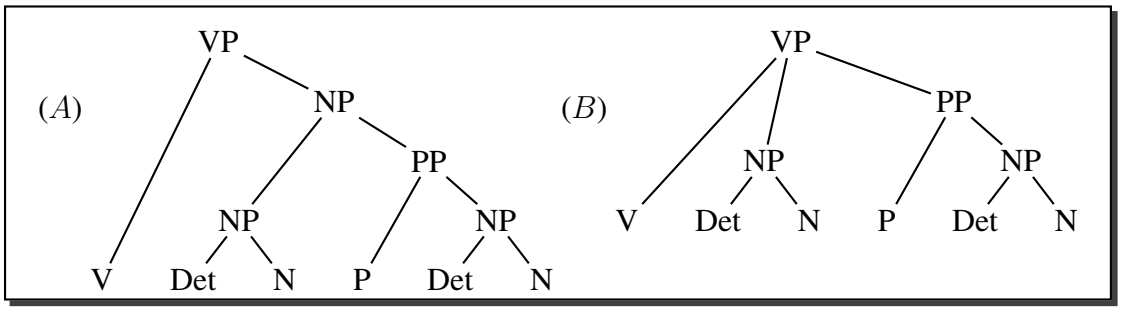

FIG. 2 – Deux analyses : (A) correspond par exemple à "*Regarde la femme avec le chapeau*", et (B) à "*Regarde la femme avec tes lunettes*".

Illustrons sur cet exemple quelques limitations des STSG pour l'analyse. Considérons par exemple le cas où aucune autre information sur le langage utilisé n'est disponible. Dans ce cas, il peut être naturel de chercher à imposer que la grammaire affecte la même probabilité à chaque analyse. Avec une STSG, l'affectation la plus "impartiale" des probabilités élémentaires semble correspondre à $p_1 = p_2 = p_3 = p_4 = \frac{1}{2};\ p_5 = 1$: tous les arbres élémentaires de même racine sont équiprobables, et la somme de leurs probabilités est 1.

En réalité, cependant, la grammaire obtenue est très peu impartiale : les contraintes stochastiques choisies favorisent ici les arbres les plus petits, i.e. ceux qui sont produits avec le moins d'étapes de dérivation. Par exemple, $p_{STSG}(d_1(A))$ et $p_{STSG}(d_1(B))$ sont respectivement $(\frac{1}{2})^4$ et $(\frac{1}{2})^3$: (B) est donc deux fois plus probable que (A) !

Avec une GTSG au contraire l'affectation impartiale $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = \lambda_5 = 0$ mène à des probabilités conditionnelles de $d_1(A)$ et $d_1(B)$ égales à $1/2$, ce qui apparaît plus conforme à l'intuition.

Notez que cet exemple a pour seul but d'illustrer la différence entre GTSG et STSG, non de démontrer la supériorité de l'une sur l'autre. Toutefois, les expérimentations pratiques confirment que des inconvénients flagrants des STSG, soulignés dans (Johnson, 1998) et (Bonnema

et al., 1999), sont effectivement évités avec les GTSG associées à la technique d'apprentissage décrite ci-après.

3 Apprentissage des paramètres à partir d'un corpus

Les paramètres du modèle sont appris à partir d'un corpus $\mathcal{C}$ selon le principe de Maximum de Vraisemblance (Dempster *et al.*, 1977).

Nous cherchons pour cela les paramètres λ^* de la GTSG qui maximisent la vraisemblance conditionnelle du corpus d'apprentissage $L_{\tilde{p}}(p_\lambda)$, où $\tilde{p}(w,t)$ est la fréquence relative dans $\mathcal{C}$ de l'arbre t de feuilles w :

$$\lambda^* = \operatorname*{Argmax}_{\lambda} \sum_{w,t} \tilde{p}(w,t) \log p_\lambda(t|w)$$

3.1 Algorithme "Improved Iterative Scaling" (IIS)

Pour résoudre ce problème non-trivial de maximisation, nous appliquons l'algorithme IIS généralisé (Lafferty, 1996) aux STSG[8] : plutôt que de maximiser $L_{\tilde{p}}(p_\lambda)$ directement, cet algorithme recherche le maximum en améliorant itérativement le modèle λ, à partir d'un modèle λ_0 initial.

Dans notre cas, cette méthode conduit à chercher pour chaque arbre élémentaire $\widehat{\tau}$ la solution $\hat{x}$ de l'équation suivante, où $f^{\#}(\tau) = \sum_{\tau \in \text{Grammaire}} f_\tau(d)$:

$$\sum_{w,t} \tilde{p}(w,t) \sum_{d \Rightarrow w} p_\lambda(d|w) f_{\widehat{\tau}}(d) x^{f^{\#}(d)} = \sum_{w,t} \tilde{p}(w,t) \sum_{d \Rightarrow t} p_\lambda(d|t) f_{\widehat{\tau}}(d) \tag{3}$$

Les modèles successifs sont alors obtenus en remplaçant $\lambda_{\widehat{\tau}}$ par $\lambda_{\widehat{\tau}} + \log \hat{x}$.

Comme ses coefficients sont positifs, l'équation (3) peut facilement se résoudre par la méthode de Newton. La difficulté résiduelle est de calculer les coefficients en un temps "raisonnable", de façon à ce que l'apprentissage puisse être effectivement réalisé. La prise en compte de cette notion d'efficacité est l'objet de la section suivante.

3.2 Algorithme Inside-Outside

Membre de gauche Le membre de gauche de la formule de réestimation (3) repose sur une double somme : l'une sur les exemples de la base, l'autre sur les dérivations possibles pour une phrase donnée.

Le calcul de cette dernière doit être factorisé pour être effectué en pratique[9].

De fait, le terme le plus problématique de la partie gauche de (3) peut se récrire :

$$\sum_{d \Rightarrow w} p_\lambda(d|w) f_{\widehat{\tau}}(d) x^{f^{\#}(d)} = \frac{1}{Z_\lambda(w)} \sum_{d \Rightarrow w} e^{\sum_{\tau \in \mathcal{R}} \lambda_\tau f_\tau(d)} f_{\widehat{\tau}}(d) x^{f^{\#}(d)}$$

$$= \frac{1}{Z_\lambda(w)} \sum_{d \Rightarrow w} f_{\widehat{\tau}}(d) x^{f^{\#}(d)} \prod_{\tau \in d} \left(e^{\lambda_\tau}\right)^{f_\tau(d)} = \frac{1}{Z_\lambda(w)} \sum_{d \Rightarrow w} f_{\widehat{\tau}}(d) \prod_{\tau \in d} v(\tau)^{f_\tau(d)}$$

[8] les variables cachées de l'article mentionné correspondent aux dérivations des arbres, qui ne sont pas explicites dans la base d'apprentissage.

[9] Le nombre de dérivations d'une analyse peut être exponentiel en sa taille, et, de plus, une phrase peut avoir un grand nombre d'analyses possibles.

où $v(\tau) = e^{\lambda_\tau} x$.

On peut alors calculer $\sum_{d \Rightarrow w} f_{\widehat{\tau}}(d) \prod_{\tau \in d} v(\tau)^{f_\tau(d)}$ en utilisant l'algorithme *Inside-Outside* décrit dans (Goodman, 1998) sur le semi-anneau des polynômes $\mathbb{P}[v(\tau)]$.

De plus, nous avons par définition : $Z_\lambda(w) = \sum_{d \Rightarrow w} \prod_{\tau \in d} \left(e^{\lambda_\tau}\right)^{f_\tau(d)}$. Il apparaît ainsi que $Z_\lambda(w)$ est la somme des coefficients associés au $\widehat{\tau}$ tel que $f_{\widehat{\tau}}(d) = 1$, qui est déjà calculée dans l'algorithme Inside-Outside. Le membre de gauche est donc complètement déterminé.

Membre de droite De façon similaire, le membre de droite de (3) peut se récrire :

$$\sum_{d \Rightarrow t} p_\lambda(d|t) f_{\widehat{\tau}}(d) = \frac{1}{Z_\lambda(t)} \sum_{d \Rightarrow t} f_{\widehat{\tau}}(d) \prod_{\tau \in d} v'(\tau)^{f_\tau(d)}$$

où $v'(\tau) = e^{\lambda_\tau}$. Là encore, $\sum_{d \Rightarrow t} f_{\widehat{\tau}}(d) \prod_{\tau \in d} v'(\tau)^{f_\tau(d)}$ peut être calculé par l'algorithme Inside-Outside, en "épurant" préalablement la table utilisée pour factoriser les calculs intermédiaires, de façon à ce que seules les dérivations menant à l'analyse t y figurent.

3.3 Apprentissage par profondeur croissante

L'algorithme IIS présente un inconvénient similaire à celui décrit dans (Bonnema & Scha, 2002) pour DOP : dans le cas où l'ensemble des arbres élémentaires $\mathcal{R}$ contient les analyses complètes du corpus d'apprentissage, le modèle sur-estime les paramètres de telle sorte que les analyses non présentes dans le corpus reçoivent une probabilité arbitrairement petite.

Pour éviter ce comportement, on peut décider de ne pas mettre les arbres complets du corpus dans $\mathcal{R}$. Une autre possibilité est de changer légèrement la méthode d'apprentissage. Plutôt que de considérer la vraisemblace du corpus avec la grammaire complète, on peut considérer, de façon itérative, les vraisemblances du corpus en introduisant progressivement des arbres élémentaires dans la grammaire. Nous avons choisi d'introduire les arbres élémentaires par ordre de profondeur croissante. Notez que ce type de mécanisme n'est rien de plus qu'une méthode de lissage de l'apprentissage.

Nous considérons donc l'ensemble des TSG suivantes : $\mathcal{G}_p$ est la grammaire associée à l'ensemble $\mathcal{R}_p$ des sous-arbres du corpus de profondeur maximale p (trivialement $\mathcal{R}_p \subset \mathcal{R}_{p+1}$), et à l'ensemble $\lambda_{\mathcal{R}_p}$ des paramètres correspondants. La méthode d'apprentissage par profondeur croissante (IDL)[10] consiste alors à calculer les paramètres $\lambda_{\mathcal{R}_1}$ de $\mathcal{G}_1$ par l'algorithme *Improved Iterative Scaling* (IIS), et pour chaque profondeur p, à calculer les paramètres $\lambda_{\mathcal{R}_p} \backslash \lambda_{\mathcal{R}_{p-1}}$ de $\mathcal{G}_p$,[11] en gardant $\lambda_{\mathcal{R}_{p-1}}$ constant. Les paramètres $\lambda_{\mathcal{R}_p} \backslash \lambda_{\mathcal{R}_{p-1}}$ sont déterminés en maximisant la probabilité conditionnelle $L_{\tilde{p}}(p_{\lambda_{\mathcal{R}_p}})$ du corpus d'apprentissage selon le modèle $\mathcal{G}_p$, *sachant* $\lambda_{\mathcal{R}_{p-1}}$.

En pratique, IDL ne nécessite qu'une adaptation minime des algorithmes précédents. Seules les étapes d'initialisation et de mise-à-jour sont modifiées ; les calculs proprement dits restent inchangés :

– Initialisation : les paramètres $\lambda_{\mathcal{R}_{pmax}} \backslash \lambda_{\mathcal{R}_p}$ valent $-\infty$; les paramètres $\lambda_{\mathcal{R}_p} \backslash \lambda_{\mathcal{R}_{p-1}}$ valent 0 ; et les paramètres $\lambda_{\mathcal{R}_{p-1}}$ gardent la valeur prise à l'itération précédente.

[10] IDL = Increasing Depth Learning.

[11] $\lambda_{\mathcal{R}_p} \backslash \lambda_{\mathcal{R}_{p-1}}$ represente les paramètres de $\lambda_{\mathcal{R}_p}$ qui ne sont pas dans $\lambda_{\mathcal{R}_{p-1}}$.

– Mise-à-jour : seuls les paramètres $\lambda_{\mathcal{R}_p}$ sont modifiés.

IIS est ainsi répété $pmax$ fois, $pmax$ étant la profondeur maximale des arbres du corpus, ce qui allonge évidemment le temps d'apprentissage[12]. Cependant, la convergence IIS étant d'autant plus rapide que les arbres considérés sont de grande profondeur, on peut encore optimiser IDL en adaptant le nombre de passes IIS à la profondeur des arbres dont on apprend les paramètres.

4 Expériences

Le principe d'apprentissage exposé repose sur la probabilité d'un corpus d'analyses conditionnellement aux phrases. Pour rester cohérent avec ce principe, l'analyse syntaxique doit reposer sur cette même probabilité. Dans cette optique, le critère de choix parmi les analyses syntaxiques possibles sera donc de sélectionner l'analyse la plus probable (MPP) parmi toutes les analyses possibles[13].

4.1 STSG polynomiales

Les STSG posent une grande difficulté : avec elles, la recherche du MPP est en effet un problème NP-difficile dans le cas général (Sima'an, 1996). Des algorithmes approximant cette recherche ont déjà été développés (Bod, 1992; Goodman, 1996; Chappelier & Rajman, 2000). Une alternative, introduite dans (Chappelier & Rajman, 2001), est d'utiliser des Grammaires à Substitution d'Arbres Polynomiales (pSTSG[14]) qui, en n'utilisant comme arbres élémentaires qu'un sous-ensemble bien choisi des sous-arbres du corpus, rendent polynomiale la complexité de la recherche du MPP. C'est le cadre choisi pour nos expériences.

Min-Max pSTSG Les pSTSG "Min-Max" sont obtenues en sélectionnant du corpus deux types d'arbres élémentaires : les sous-arbres minimaux, de profondeur 1, et les sous-arbres maximaux, i.e. dont toutes les feuilles sont des terminaux. La polynomialité des pSTSG Min-Max a été prouvée dans (Chappelier & Rajman, 2001).

Head-driven pSTSG Les "Head-driven pSTSG" sont obtenues de façon similaire : tous les sous-arbres de profondeur 1 sont sélectionnés du corpus, les autres arbres élémentaires étant ceux dont la "branche de tête" est étendue autant que possible. Une "branche de tête" est une branche dont tous les nœuds, associés à des catégories syntaxiques, sont étendus seulement s'ils correspondent à la tête lexicale du nœud qui les domine. Les têtes lexicales sont définies comme dans (Collins, 1999).

4.2 Protocole

La version de Bod du corpus ATIS a été utilisée pour cette évaluation. Elle consiste en 750 arbres syntaxiques dans lesquels les feuilles lexicales ont été supprimées. La Min-Max pSTSG extraite compte 2 434 arbres élémentaires : 381 sont de profondeur 1, les autres étant des arbres

[12] Un ou deux jours sur une Sun Sparc 10, pour un corpus d'apprentissage de 3000 arbres, $pmax = 16$, et 200 passes IIS par profondeur.

[13] D'autres approches sélectionnent la *dérivation la plus probable*, ou l'analyse ayant le *maximum de constituants probablement justes* – voir (Goodman, 1996).

[14] pSTSG = Polynomial Stochastic Tree Substitution Grammars.

maximaux. La Head-driven pSTSG compte seulement 930 arbres élémentaires, dont 381 de profondeur 1, et 549 “branches de têtes”.

La profondeur maximale des arbres du corpus est de 11. C’est donc également la profondeur maximale des sous-arbres du modèle Min-Max. En revanche, la profondeur maximale est de 5 pour le modèle Head-driven, certaines règles n’ayant pas de tête lexicale, et les “branches de têtes” n’étant pas forcément les plus longues dans un arbre.

Deux types d’expériences ont été réalisés. Dans “test 10%”, nous entraînons les modèles sur 90% du corpus, et nous les testons sur les 10% restant. Les résultats correspondent aux moyennes des valeurs obtenues pour dix partitionnements aléatoires. Dans “Autotest”, apprentissage et test sont réalisés sur le corpus complet ; ces secondes expériences ne servent donc que de repère sur le gain maximum que l’on peut espérer par l’utilisation des modèles gibbsiens.

Dans chaque expérience, le modèle GTSG est comparé à une STSG obtenue par la méthode d’apprentissage standard, où les arbres élémentaires reçoivent des probabilités proportionnelles à leur fréquence dans le corpus.

Pour les modèles Head-driven GCFG, deux types d’apprentissage on été testés sur la partie “test 10%” : le premier utilise la procédure IDL, la seconde non. De fait, cette procédure peut être évitée pour les Head-driven GTSG, car les arbres complets du corpus ne font pas partie de l’ensemble de leurs arbres élémentaires. De même, IDL est inutile dans la partie “Autotest”, car on ne cherche pas alors à observer les capacités de généralisation des grammaires.

Pour évaluer les résultats, nous avons utilisé les scores standards définis dans PARSEVAL (Manning & Schütze, 1999) :
– “*Couverture*” : nombre de phrase recevant au moins une analyse (correcte ou pas) ;
– “*Correcte*” : taux de phrases correctement analysées parmi les phrases couvertes.
– “*Croisé*” : taux de phrases recevant un parenthésage incorrect parmi les phrases couvertes. Un parenthésage est incorrect lorsqu’un des groupes syntaxiques d’une analyse a une intersection avec un groupe de l’analyse de référence, sans que l’un des deux groupes soit inclus dans l’autre.
– *Précision* “P” et *Rappel* “R” sont obtenus en considérant la séquence d’arbres $\tilde{\tau}$ produits par l’analyseur comme un ensemble $E(\tilde{\tau})$ de triplets $< N, p, d >$, où N est un label syntaxique attaché à un nœud d’un arbre, et p et d sont les index dans le corpus des premier et dernier mots dominés par N. Par comparaison avec la séquence des arbres de référence $\tilde{\tau}'$, les taux sont calculés comme :

$$P(\tilde{\tau}) = \frac{|E(\tilde{\tau}) \cap E(\tilde{\tau}')|}{|E(\tilde{\tau})|},\ R(\tilde{\tau}) = \frac{|E(\tilde{\tau}) \cap E(\tilde{\tau}')|}{|E(\tilde{\tau}')|}$$

– le *f-score* “F” est la moyenne harmonique de la précision et du rappel :

$$F^{-1} = \frac{1}{2}(P^{-1} + R^{-1})$$

.

4.3 Résultats

Les résultats obtenus pour les pSTSG Min-Max et Head-driven sont résumés dans la table 1.

Avec les corpus mentionnés, le processus d’apprentissage prend environ 4 heures pour chaque modèle sur une Sun Blade 60. Pour la procédure IDL, 20 itérations de IIS ont été réalisées pour chaque profondeur d’arbre élémentaire. Sans IDL, 200 itérations IIS ont été utilisées pour l’ensemble complet des arbres élémentaires.

Min-max							Head-Driven						
Autotest							**Autotest**						
	Couv.	Corr.	Croisé	P	R	F		Couv.	Correct	Croisé	P	R	F
STSG	750	664	0				STSG	750	412	57			
Taux (en %)		88.5	0.0	99.7	99.2	99.45	Taux (en %)		54.9	7.6	96.9	95.2	96.0
GTSG	750	691	0				GTSG	750	479	43			
Taux (en %)		92.1	0.0	99.6	99.5		Taux (en %)		63.8	5.7	97.6	97.0	97.3
Gain (%)		**3.6**		–0.1	0.2	0.05	Gain (%)		**8.9**	–1.9	0.7	1.8	1.25
Test 10%							**Test 10%**						
STSG	73.9	35.6	11.6				STSG	73.9	30.4	10.4			
Taux (en %)		48.2	15.7	93.6	94.4	94.0	Taux (en %)		41.1	14.0	93.7	93.6	93.6
GTSG-IDL	73.9	36.4	9.7				GTSG-IIS	73.9	35.7	10.0			
Taux (en %)		49.3	13.1	93.5	95.6	94.5	Taux (en %)		48.3	13.5	94.2	95.1	94.6
Gain (%)		**1.1**	–2.6	–0.1	1.2	0.5	Gain (%)		**7.2**	0.5	0.5	1.5	1
							GTSG-IDL	73.9	35.2	10.2			
							Taux (en %)		47.6	13.8	94.2	94.8	94.4
							Gain (%)		**6.5**	0.2	0.5	1.2	0.8

TAB. 1 – Performances en analyse : GTSG comparées aux STSG.

4.4 Discussion

Les GTSG obtiennent le score maximum en autotest pour les grammaires Min-Max[15]. C'est une illustration du fait que les faiblesses théoriques soulignées dans (Bonnema *et al.*, 1999) affectent les STSG et non les GTSG. La suppression de ces faiblesses n'a presque aucun effet en "test-10%". Nous pensons que ces effets sont masqués par les autres facteurs d'erreur qui agissent dans cette situation, comme la grande variabilité du corpus utilisé.

L'avantage des Head-driven GTSG sur les Head-driven STSG est plus évident. Le taux d'analyses fausses est réduit de 12% sur les expériences "test-10%". Les taux de parenthésage croisé, de précision et de rappel en label sont également meilleurs.

Comme prévu par la théorie, la procédure IDL est inutile avec les GTSG Head-driven : les modèles obtenus avec un IIS standard mènent à de meilleurs résultats.

5 Conclusion

Pour conclure, les modèles log-linéaires associés à un apprentissage discriminant développés dans ce document semblent d'autant plus intéressants que les grammaires comptent moins de paramètres : les grammaires Head-driven en bénéficient plus que les modèles Min-max.

Les résultats obtenus en testant les modèles sur le corpus d'apprentissage semblent confirmer que les "faiblesses" théoriques connues des STSG sont supprimées dans les GTSG. Cet avantage est dû principalement à la fonction d'apprentissage discriminante et conditionnée par les feuilles que l'on utilise dans le modèle GTSG, qui est mieux adaptée à la tâche d'analyse syntaxique à laquelle on destine la grammaire. L'amélioration est cependant moins sensible en généralisation, du fait de l'insuffisance des données d'apprentissage par rapport à la taille de la grammaire.

Les tests menés ne montrent cependant pas l'avantage que l'on pourrait tirer d'une autre caractéristique des GTSG, à savoir l'absence de normalisation de leurs paramètres. Nous pressentons en effet que cette absence pourrait être bénéfique dans des applications telles que la

[15]Le score maximal n'est pas 100%, du fait que le corpus compte 555 arbres *différents* sur ses 750, pour seulement 512 phrases *différentes*, du fait de la délexicalisation appliquée.

reconnaissance de la parole, où la nature probabiliste des paramètres pose des problèmes lors du mélange de modèles (modèle acoustique et modèle de langage). Il serait très intéressant d'instancier les GTSG, avec une procédure d'apprentissage adaptée, dans un tel contexte.

Références

BOD R. (1992). Applying Monte Carlo techniques to Data Oriented Parsing. In *Proceedings Computational Linguistics in the Netherlands*, Tilburg (The Netherlands).

BOD R. (1998). *Beyond Grammar, An Experience-Based Theory of Language*. Number 88 in CSLI Lecture Notes. Standford (CA) : CSLI Publications.

BONNEMA R., BUYING P. & SCHA R. (1999). A new probability model for data oriented parsing. In P.DEKKER & G.KERDILES, Eds., *Proceedings of the 12th Amsterdam Colloquium*, Amsterdam : Institute for Logic, Language and Computation.

BONNEMA R. & SCHA R. (2002). Reconsidering the probability model of data-oriented parsing. In R. BOD, R. SCHA & K. SIMA'AN, Eds., *Data-Oriented Parsing*, chapter I.3, p. 25–41. CSLI Publications.

CHAPPELIER J.-C. & RAJMAN M. (2000). Monte-Carlo sampling for NP-hard maximization problems in the framework of weighted parsing. In D. CHRISTODOULAKIS, Ed., *Natural Language Processing – NLP 2000*, number 1835 in Lecture Notes in Artificial Intelligence, p. 106–117. Springer.

CHAPPELIER J.-C. & RAJMAN M. (2001). Grammaire à substitution d'arbre de complexité polynomiale : un cadre efficace pour DOP. In *TALN'2001*, volume 1, p. 133–142.

COLLINS M. (1999). *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania.

COLLINS M. & DUFFY N. (2002). New ranking algorithms for parsing and tagging : Kernels over discrete structures, and the voted perceptron. In *ACL2002*, p. 263–270.

DEMPSTER M. M., LAIRD N. M. & JAIN D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistics Society*, **39**, 1–38.

GOODMAN J. (1996). Efficient algorithms for parsing the DOP model. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing*, p. 143–152.

GOODMAN J. (1998). *Parsing Inside-Out*. PhD thesis, Harvard University. cmp-lg/9805007.

JOHNSON M. (1998). PCFG Models of Linguistic Tree Representations. *Computational Linguistics*, **24**(4), 613–632.

LAFFERTY J. (1996). Gibbs-Markov models. In *Computing Science and Statistics*, volume 27, p. 370–377.

MANNING C. & SCHÜTZE H. (1999). *Foundations of Statistical Natural Language Processing*. Cambridge : The MIT Press.

ROZENKNOP A. (2002). Une grammaire hors-contexte valuée pour l'analyse syntaxique. In *9ème Conférence Annuelle sur le Traitement Automatique des Langues Naturelles*, volume 1, p. 95–104, Nancy : Association pour le Traitement Automatique des Langues (ATALA).

SIMA'AN K. (1996). Computational complexity of probabilistic disambiguation by means of tree grammars. In *Proceedings of COLING'96*, Copenhagen (Denmark). cmp-lg/9606019.