DÉTECTION DE GROUPES NOMINAUX PAR INFÉRENCE GRAMMATICALE PROBABILISTE

Franck Thollard* Alexander Clark†

* EURISE, Université de Saint-Etienne, France
thollard@univ-st-etienne.fr

† ISSCO/TIM, Université de Genève, Suisse
alex.clark@issco.unige.ch

Résumé

Nous présentons dans cet article l'utilisation de l'inférence grammaticale probabiliste pour l'analyse syntaxique partielle. Nous décrivons dans un premier temps l'utilisation d'un automate probabiliste qui modèle la probabilité jointe de groupes syntaxiques et de leur parties de discours correspondantes. Nous considérons ensuite l'intégration efficace de l'information lexicale dans le modèle. Enfin, une étude de l'adaptation d'une technique de ré-échantillonage (le *bagging*) à notre cadre probabiliste est proposée.

Mots Clés : Inférence grammaticale probabiliste, Analyse syntaxique, Bagging

1 INTRODUCTION

L'analyse syntaxique partielle d'une langue naturelle consiste à décomposer les phrases en séquences de syntagmes simples. Ces techniques sont, entre autre, utilisées comme premières étapes pour l'analyse syntaxique complète de phrases.

Dans cette tâche l'algorithme prend en entrée un mot et la partie du discours (aussi appelée étiquette grammaticale) qui lui correspond. L'objectif est de prédire le syntagme (syntagme verbal, syntagme nominal,...) de ce couple. On ne cherche pas ici à décomposer les syntagmes (comme par exemple identifier les compléments).

Le présent travail étend l'étude préliminaire réalisée dans (Thollard, 2001b) en proposant un méthode permettant la prise en compte efficace et automatique de l'information lexicale et l'adaptation du *bagging* (Breiman, 1996) à notre cadre.

Nous décrivons dans un premier temps les données et le protocole utilisé. Nous présentons ensuite plus en détail l'intégration de l'information lexicale au modèle ainsi que l'adaptation de la technique d'échantillonnage appelée *bagging*.

2 LE PROBLÈME

Le problème consiste à construire un parenthésage de phrases de langue naturelle : nous recherchons les bornes des syntagmes les plus larges. La langue considérée ici est l'anglais. Cependant, d'autre langues pourraient être traitées sans modification des techniques pourvu qu'un ensemble d'apprentissage puisse être fourni à la méthode. Pour la phrase *"He reckons the current account deficit will narrow to only # 1.8 billion in September."*, le but sera de définir le parenthésage suivant[1] *[NP He] [VP reckons] [NP the current account deficit] [VP will narrow] [PP to] [NP only # 1.8 billion] [PP in] [NP September] .*

Les données utilisées proviennent de textes du journal "Wall Street Journal" qui ont été annotés syntaxiquement (Marcus *et al.*, 1993). Les syntagmes les plus fréquents sont les syntagmes nominaux (51%), verbaux (20%) et propositionnels (20%). La tâche consiste à identifier le syntagme étant donné les mots précédents et leurs parties de discours (notée *PdD* en abrégé). Afin de mieux simuler la réalité, les parties de discours utilisées sont obtenues par un étiqueteur syntaxiques et ne sont pas celles du corpus. L'étiqueteur syntaxique utilisé pour réaliser cette tâche est l'étiqueteur de Brill[2] (1995).

L'interprétation de l'étiquette du syntagme est la suivante : pour le syntagme C, B-C signifiera "Début du syntagme C" et I-C "Dans le syntagme C". Le début d'un syntagme donné fermera automatiquement le syntagme précédent.

Pour fixer les idées, nous donnons l'exemple suivant :

He	PRP	B-NP	....		
reckons	VBZ	B-VP	billion	CD	I-NP
the	DT	B-NP	in	IN	B-PP
current	JJ	I-NP	Sept	NNP	B-NP
....			.	.	O

La première colonne représente les mots, la deuxième représente les parties du discours (*PdD*) obtenues par l'étiqueteur de Brill et la dernière les syntagmes extraits du corpus.

3 LE POINT DE VUE

Le point de vue adopté ici est le même que celui de l'étude (Thollard, 2001b). Nous ne la rappelons que brièvement ici. Notre objectif est de traiter les syntagmes, c'est à dire d'identifier un syntagme étant donné les parties du discours précédentes. On cherche alors à trouver l'ensemble de syntagmes 1 à n le plus probable étant donné les parties du discours. Si $T_{1,n}$ (resp. $C_{1,n}$) représente les *PdD* 1 à n (resp. le

1. NP provient de la terminologie anglaise et signifie "syntagme nominal" (abréviation de Noun Phrase). De même, VP signifie syntagme verbal et PP syntagme propositionnel.

2. Un étiqueteur syntaxique assigne une *PdD* à un mot : le mot "ferme" se verra attribuer la *PdD* `nom` dans le contexte "la ferme est belle" et `verbe` dans le contexte "il ferme la porte".

syntagme 1 à n), on cherche à trouver les syntagmes $S_{1,n}$ qui satisfont :

$$S_{1,n} = \arg\max_{C_{1,n}} P(C_{1,n}|T_{1,n}) = \arg\max_{C_{1,n}} \frac{P(C_{1,n},T_{1,n})}{P(T_{1,n})} = \arg\max_{C_{1,n}} P(C_{1,n},T_{1,n})$$

Le problème est alors d'estimer la probabilité jointe $P(C_{1,n},T_{1,n})$ dont une décomposition est :

$$\begin{aligned} P_{1,n} &= P(T_1)\, P(C_1|T_1)\, P(T_2|C_1,T_1)\, P(C_2|T_{1,2},C_1) \times \\ & \quad \cdots \times P(T_n|T_{1,n-1},C_{1,n-1})\, P(C_n|T_{1,n},C_{1,n-1}) \\ &= \prod_{i=1}^{n} P(C_i,T_i|C_{1,i-1},T_{1,i-1}) \end{aligned}$$

Nous voulons donc déterminer le syntagme le plus probable étant donné l'historique de paires (*PdD*,syntagmes). Nous pouvons représenter ces paires en concaténant les parties du discours et les syntagmes en un seul symbole. Si on utilise le symbole "+" comme séparateur, l'exemple précédent ressemblera à : `PRP+B-NP VBZ+B-VP DT+B-NP ...IN+B-PP NNP+B-NP .+O`

L'algorithme utilisé ici sera l'algorithme DDSM (Thollard, 2001a) car contrairement à la plupart de ses concurrents (parmi lesquels on trouve les n-grammes et leurs techniques de lissage associées (Abe & K.Warmuth, 1992; Goodman, 2001), et les modèles de Markov à dépendance variable (Schütze & Singer, 1994; Ron *et al.*, 1995)), il permet d'inférer des modèles dont les probabilités sont estimées avec un historique non borné. D'autres algorithmes inférant des modèles à dépendance quelconques existes mais n'ont pas été retenus car trop lourds calculatoirement (Stolcke & Omohundro,) ou moins performants (Carrasco & Oncina, 1994).

Nous présentons maintenant formellement le modèle et l'algorithme d'inférence DDSM.

4 L'ALGORITHME D'APPRENTISSAGE

Nous présentons dans un premier temps la définition formelle du modèle ; vient ensuite la présentation de l'algorithme d'inférence.

Un *Automate Fini Déterministe Probabiliste* (PDFA) est un 7-tuple $(\Sigma, Q, q_I, \xi, \delta, \gamma, F)$ où Σ est l'alphabet, Q est l'ensemble des états, $q_I \in Q$ est l'état *initial*, $\xi \subset Q \times \Sigma \times Q \times$ (0,1] est un ensemble de transitions probabilistes. F: $Q \to$ [0,1] est la probabilité que l'automate termine dans un état particulier. δ de $Q \times \Sigma$ dans Q est la fonction de transition déterministe telle que $\delta(q_i,\sigma) = q_j$ si il existe p tel que $(q_i,\sigma,q_j,p) \in \xi$. De même, γ de $Q \times \Sigma$ dans $(0,1]$ est telle que $\gamma(q_i,\sigma) = p$ si il existe $q_j \in Q$ tel que $(q_i,\sigma,q_j,p) \in \xi$.

De plus, si pour chaque état q, $\sum_\sigma \xi^A(q,\sigma,q') + F^A(q) = 1$ et q peut générer au moins une chaîne de probabilité strictement positive alors l'automate définit une distribution de probabilités sur $\Sigma^\star$.

L'algorithme DDSM s'inscrit dans le schéma classique d'inférence grammaticale probabiliste. Il construit dans un premier temps un automate qui représente le maximum de vraisemblance des données (le PPTA) puis le généralise par une opération de fusion d'états.

Soit I_+ un *ensemble positif*, *i.e.* un ensemble de chaînes appartenant au langage probabiliste que l'on veut modéliser. Soit $PTA(I_+)$ l'*arbre accepteur des préfixes* construit à partir de I_+. L'*arbre accepteur des préfixes* est un automate qui accepte uniquement les chaînes de I_+ et dans lequel les préfixes communs ont été fusionnés, engendrant une structure d'arbre. Le $PPTA(I_+)$ est l'extension probabiliste du $PTA(I_+)$: soit $C(q)$ le compte de l'état q, c'est à dire, le nombre de fois où l'état q a été utilisée pour générer I_+ à partir du $PPTA(I_+)$, et $C(q,\#)$ le nombre de fois où l'analyse d'une chaîne de I_+ finit en q. Soit, $C(q,a)$ le compte de la transition $(q,a,\bullet,\bullet)$ dans le $PPTA(I_+)$. L'estimation de la probabilité d'une transition s'exprime comme : $\hat{\gamma}(q,a) = \frac{C(q,a)}{C(q)}$, $a \in \Sigma \cup \{\#\}$. Le $PPTA(I_+)$ est alors l'estimation du maximum de vraisemblance de la distribution empirique construite à partir de I_+. Un exemple de $PPTA$ est présenté en figure 1.

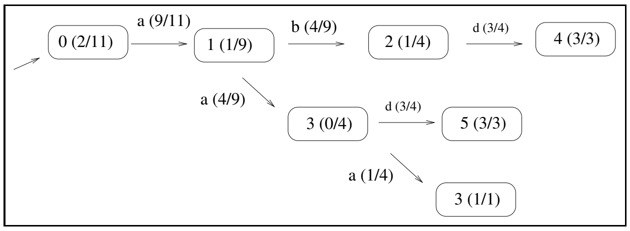


FIG. 1 – *PPTA construit avec $I+$ = {λ, aac, aac, abd, aac, aac, abd, abd, λ, a, ab}*

Nous présentons maintenant le deuxième outil utilisé par l'algorithme DDSM : la fusion d'états. Cette opération produit deux modifications au sein de l'automate : (i) elle en modifie la structure (voir figure 2, partie gauche) et (ii) les probabilités (voir figure 2, partie droite). Elle s'applique à deux états. La fusion de deux états peut mener à un non déterminisme ; les états qui créent ce non-déterminisme sont alors récursivement fusionnés. La mise à jour des probabilités pour un état q issu de la fusion des états q' et q'' respecte l'égalité suivante :

$$\gamma(q,a) = \frac{C(q',a)+C(q'',a)}{C(q')+C(q'')} \quad ,\forall a \in \Sigma \cup \{\#\}$$

Nous disposons maintenant des outils nécessaires à la présentation de l'algorithme DDSM.

4.1 L'algorithme DDSM

L'algorithme DDSM (Thollard, 2001a) (algorithme 1) prend deux arguments en paramètre : l'ensemble d'apprentissage I_+ et un paramètre de réglage α. Il cherche

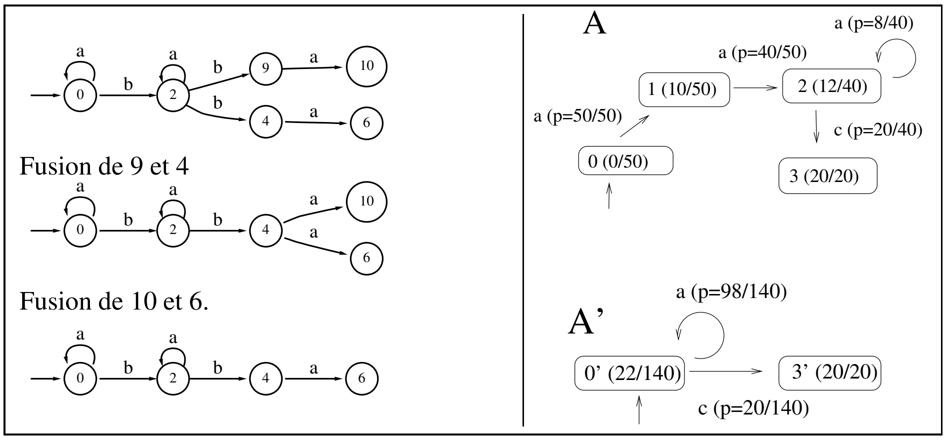


FIG. 2 – *Fusion* .

à inférer un automate qui est un compromis entre une distance (mesurée par la divergence de Kullback-Leibler (Cover & Thomas, 1991)) petite par rapport aux données et une faible taille (mesurée par le nombre d'états). Les données sont représentées par le PPTA qui est l'estimation du maximum de vraisemblance des données. Sauf cas particulier, une fusion augmente la distance entre l'automate courant et les données et amène, dans le même temps, une diminution de la taille du modèle. Deux états sont dits compatibles si l'impact en termes de divergence de leur fusion, divisé par le gain en taille est inférieur au paramètre α.

L'algorithme DDSM construit dans un premier temps le $PPTA(I_+)$ et itère ensuite les opérations suivantes : choisir deux états (les états q_i et q_j dans le code), et calculer la valeur de leur compatibilité (Divergence_Relative dans le code) impliquée par la fusion. Il réalise ensuite la meilleure fusion si elle est compatible. L'ordre utilisé est une adaptation de celui de l'algorithme EDSM qui a gagné une compétition d'inférence grammaticale non probabiliste (Lang *et al.*, 1998).

Le paramètre α permet d'un certain point de vue de contrôler le degré de généralisation de l'algorithme. En effet, en général, plus la valeur de α est grande, plus général et petit est l'automate inféré.

5 AUTOMATES VUS COMME DES TRANSDUCTEURS

Nous décrivons dans cette section comment l'automate peut être considéré comme un transducteur totalement aligné, c'est à dire, un transducteur qui émet un symbole à chaque symbole lu en entrée. L'automate de la figure 3 peut être vu comme un transducteur de même structure qui émet les symboles sur les transitions. Il prend en entrée les *PdD* (par exemple *NN*) et émet les syntagmes correspondants (par exemple *B-NP*). Ces transducteurs sont non-subséquentiels car non-déterministes

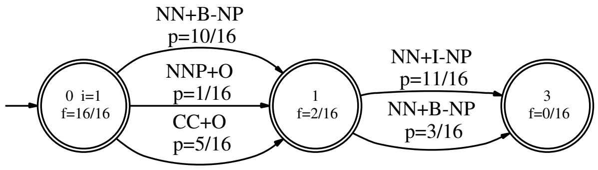
Algorithme 1: DDSM ($I+, \alpha$).

Algorithme :

A ← `Numrérotation_largeur_d_abord`(*PPTA*) ;
pour $q_i = 1$ *to* `Nb_Etats` *(A)* **faire**
  best_div = ∞ ;
  pour $q_j = 0$ *à* $i - 1$ **faire**
    si `Compatible`$(A, q_i, q_j) <$ *best_div* **alors**
      best_pred = q_j ;
      best_div = `Compatible` (A, q_i, q_j);
  si *best_div* $< \alpha$ **alors** `Fusion` (A,q_i,best_pred) ;
`Retourne` A ;

par rapport à leurs entrées. Sur la figure 3 le non déterminisme s'exprime au niveau de l'état 1.

FIG. 3 – *Automate/Transducteur probabiliste*



Nous pouvons dès lors utiliser l'algorithme de Viterbi (Jurafsky & Martin, 2000) pour trouver le chemin le plus probable étant donné la séquence de *PdD* donnée en entrée ; par exemple l'analyse de la séquence `NN NN` par l'automate de la figure 3 donnera `B-NP I-NP`.

6 LE PROTOCOLE EXPÉRIMENTAL

Le critère de qualité utilisé est la fonction F basée sur la précision et le rappel (van Rijsbergen, 1975). Pour un syntagme donné C, le rappel est le nombre de fois où C est correctement proposé divisé par le nombre de fois où il doit l'être. Par ailleurs, la précision mesure le nombre de fois où C est correctement attribué divisé par le nombre de fois où il est proposé. La combinaison de ces deux critères est classiquement réalisée par la fonction F_β :

$$F_\beta = \frac{(\beta^2 + 1).Rappel.Precision}{\beta^2.Rappel + Precision}$$

On utilise ici la fonction F avec $\beta = 1$ afin de donner le même poids à la précision et au rappel. C'est cette fonction que nous allons optimiser.

Le corpus original est divisé en deux ensembles : l'ensemble d'apprentissage (AppTot) (avec 8936 phrases et 211727 symboles), et un ensemble de test (Test) contenant 2012 phrases et 47377 symboles[3].

Nous avons divisé de manière aléatoire l'ensemble d'apprentissage afin d'obtenir deux sous-ensembles : un ensemble d'apprentissage pour la validation (AppV, 8092 phrases, 191683 symboles) et un ensemble de validation (Valid, 844 phrases, 20044 symboles). AppV contient ainsi environ 90% des phrases de AppTot. Le nombre moyen de mots par phrase est à peu près 24. La taille du vocabulaire (dans notre cas les paires de *PdD* et de syntagme) est de l'ordre de 300. Le PPTA contient aux alentour de 160.000 états. Les temps de réponses s'échelonnent[4] entre 21 minutes et 10 heures en fonction de la valeur du paramètre α utilisé. La place mémoire nécessaire à DDSM pour traiter ces données est de l'ordre de 30 Mo.

Les différents paramètres ont été estimés sur le corpus (AppV, Valid). Une inférence finale a été faite sur l'ensemble AppTot avec les valeurs des paramètres déterminées dans la phase précédente ; le modèle ainsi obtenu est évalué sur l'ensemble de test.

7 ANALYSE SUR L'ENSEMBLE DE VALIDATION

Nous décrivons dans cette section l'adaptation du bagging à notre cadre suivi de l'intégration de l'information lexicale.

7.1 Bagging d'un estimateur de distributions de probabilités

La technique de bagging (Breiman, 1996) a été utilisée avec succès dans plusieurs domaines mais ne semblait pas aider pour cette tâche particulière (Sang, 2000). Cependant, il est reporté dans (Hederson & Brill, 2000) que cette technique améliore les résultats de l'analyseur syntaxique statistique de Collins (1997) . Enfin, l'interpolation de PDFA améliore significativement les performances de ces derniers (Thollard, 2001a). Nous avons donc adapté le bagging.

La technique du bagging échantillonne l'ensemble d'apprentissage en ensembles de même taille. Nous désignerons ces ensembles par B-ensembles. Les B-ensembles suivent donc le même format que celui de l'ensemble d'apprentissage. Un modèle est alors inféré avec chaque B-ensemble. On combine ensuite, par vote majoritaire, les résultats des modèles ainsi obtenus.

Si on impose que le nombre de chaînes des B-ensembles soit le même que celui de l'ensemble d'apprentissage, les chaînes de faible probabilité ont peu de chance d'apparaître dans les B-ensembles. Nous avons donc échantillonné l'ensemble d'apprentissage jusqu'à ce que le PPTA construit sur les B-ensembles ait un nombre

3. Pour plus d'informations concernant ces données (leur construction, leur obtention,...) on pourra se référer à (Sang & Buchholz, 2000).

4. Les valeurs mentionnées ici correspondent aux temps cpu renvoyés par la commande `time`. La machine est un PC linux avec un processeur 1,8 Ghz et 512 Mo de RAM.

d'états cohérent avec celui du PPTA original. Ainsi, les chaînes qui apparaissent peu souvent dans l'ensemble d'apprentissage apparaîtront tout de même dans les B-ensembles et les chaînes qui apparaissent souvent dans l'ensemble initial apparaîtrons très souvent dans les B-ensembles.

Lors de l'utilisation du bagging sur les ensembles d'apprentissage lexicalisés, nous avons dû encore augmenter la taille des B-ensembles jusqu'à 130.000 chaînes étant donné que les performances sur les ensembles de validation continuent de s'améliorer. On notera que l'augmentation de la taille des B-ensembles ne change en rien le temps de réponse de l'algorithme puisque que la taille du PPTA reste inchangée.

La section suivante propose l'inclusion de l'information lexicale dans le modèle, principale limitation relevée dans l'étude (Thollard, 2001b).

7.2 Prise en compte de l'information lexicale

L'information lexicale est connue pour être importante. Pla & *al.* (2000) utilisent des machines statistiques à états finis et améliorent leurs résultats de 4% (la fonction $F_{\beta=1}$ passant de 86 à 90), en incluant l'information lexicale. Ils choisissent certaines *PdD* T_i et les remplacent par le symbole W_iT_i lorsque le mot à étiqueter est W_i. L'idée de cette méthode est de résoudre l'ambiguïté d'une *PdD* lorsque le mot peut y aider. Par exemple, le mot *against* indique toujours un début de groupe propositionnel (syntagme B-PP) alors que sa *PdD* (IN) est ambiguë puisqu'elle peut prédire un début de proposition ou de complément (B-SBAR). L'ajout d'une nouvelle *PdD* *against-IN* va nous permettre de lever cette ambiguïté lorsque le mot est *against*. Le problème réside maintenant dans le choix des *PdD* à spécialiser. L'approche naïve consistant à spécialiser chaque *PdD* ne fonctionne pas car (i) elle est calculatoirement trop lourde et (ii) elle mène à diminuer l'information statistique disponible pour chaque élément. L'objectif sera donc de déterminer automatiquement les *PdD* à diviser. Nous utilisons pour ce faire une technique basée sur l'information mutuelle (Cover & Thomas, 1991).

Choix de la partie du discours

Nous voulons dans un premier temps identifier la *PdD* qui sera la mieux désambiguïsée par l'information lexicale. Pour ces *PdD*, nous devrions constater qu'il existe une grande dépendance entre le mot et le syntagme. L'information mutuelle entre deux variables aléatoires mesure cette notion.

L'information mutuelle entre entre deux évènements e_1 et e_2 mesure leur interdépendance. Deux évènements sont indépendants si leur probabilité jointe est égale au produit de leurs probabilités. Ainsi, la quantité $\log \frac{P(e_1,e_2)}{P(e_1)P(e_2)}$ sera nulle si les évènements sont indépendants, positive s'ils ont plus de chance d'apparaître en même temps que de manière aléatoire et négative sinon. L'extension à des variables aléatoires se fait en sommant sur les valeurs possibles que peuvent prendre lesdites variables :

$$I(X;Y) = \sum_{X=x,Y=y} p(X=x,Y=y) \log \frac{P(X=x,Y=y)}{P(X=x)P(Y=y)} \qquad (1)$$

Étant donné une *PdD* t nous calculons l'information mutuelle entre la variable représentant le syntagme et celle représentant le mot :

$$I(C|T=t;W|T=t) = H(C|T=t) - H(C|W,T=t) \qquad (2)$$

Nous sélectionnons ensuite la *PdD* qui maximise cette quantité. Ce sera donc la partie du discours pour laquelle la connaissance du mot amène la plus grande réduction de l'incertitude du syntagme.

Toutes les probabilités seront estimées en utilisant le maximum de vraisemblance. Afin d'utiliser des informations statistiques relativement fiables, nous n'avons pas pris en compte les combinaisons rares de mot/syntagme, *i.e.* les combinaisons qui apparaissent moins de six fois.

Découpage des parties de discours

La partie de discours étant choisie, nous devons créer un nouvel élément, combinaison d'un mot et d'une *PdD*, en fonction de la capacité d'un mot à prédire un syntagme. Nous allons ici aussi utiliser un critère basé sur la théorie de l'information : soit W_t l'ensemble des mots qui ont t comme *PdD*. Soit Π^t l'ensemble des partitions de W_t. Nous aimerions trouver la partition $\pi^t \in \Pi^t$ qui mène à la plus grande réduction de l'incertitude du syntagme[5]. Le parcours exhaustif de l'ensemble des partitions n'est pas réalisable calculatoirement. Nous nous limitons donc à certaines partitions : si nous définissons $\pi_i^t = \{\{w_i\}, \{w_1, ..., w_{i-1}, w_{i+1}, ..., w_n\}\}$, nous pouvons chercher la partition π_i^t qui mène à une entropie minimale pour le syntagme étant donné la partition :

$$\pi_i^t = \arg\min_{\pi_j^t} H(C|\pi_j^t) = \arg\min_{\pi_j^t} - \sum_{s\in\pi_j^t} \sum_{c_i\in C} p(c_i, s) \log p(c_i|s)$$

Nous répétons ensuite la procédure jusqu'à obtention d'un nombre de divisions correct. L'analyse de la relation entre le nombre de divisions et la valeur de la fonction $F_{\beta=1}$ obtenue sur l'ensemble de validation montre que le nombre optimal de divisions se situe autour de 100. Cela signifie qu'un faible nombre de divisions de *PdD* est nécessaire pour améliorer les performances. Ceci est important dans la mesure où le temps de réponse de l'algorithme d'inférence n'est pas significativement affecté.

Nous avons également testé une stratégie plus globale d'optimisation dans laquelle nous cherchions une partition optimale de l'ensemble des paires <mot,*PdD*>. Les expérimentations préliminaires n'étant pas concluantes, nous n'avons pas poursuivi nos investigations.

---

5. Nous voulons en fait créer un nouvel item <mot,*PdD*>. Comme la *PdD* est fixée ici, nous devons simplement partitionner l'ensemble des mots.

8 RÉSULTATS SUR L'ENSEMBLE DE TEST

Cette section décrit les résultats obtenus sur l'ensemble de test. L'approche naïve (qui choisit le syntagme le plus probabile étant donné la *PdD*) obtient une valeur de $F_{\beta=1}$ de 77. Les résultats sont résumés dans la table 1. Ils sont obtenus sur l'ensemble de test après une inférence sur l'ensemble AppTot avec la valeur du paramètre α optimale sur l'ensemble de validation.

La première colonne donne les syntagmes; les autres colonnes donnent les scores (au sens $F_{\beta=1}$) obtenus lorsque l'on utilise l'algorithme de Viterbi initial sans utilisation de l'information lexicale (colonne I), lorsque l'on ajoute l'information lexicale en divisant les *PdD* 100 fois (colonne L), lorsque l'on utilise la technique de bagging seule (colonne B) ou avec l'information lexicale (colonne LB).

Les deux dernières colonnes donnent les résultats d'autres techniques comparables, *i.e.* un modèle de n-grammes sans information lexicale (Johansson, 2000) et un modèle de Markov caché lexicalisé (Pla *et al.*, 2000).

TAB. 1 – *Résultats en termes de $F_{\beta=1}$ sur l'ensemble de test*

syntagme	I	L	B	LB	ngrm	LMM
ADJP	48	49	58	63	55	70
ADVP	63	69	69	74	70	77
INTJ	33	22	40	67	40	100
NP	85	86	89	89	89	90
PP	87	95	90	95	92	96
PRT	9	55	20	67	32	67
SBAR	15	77	16	81	41	79
VP	86	86	89	88	90	92
ALL	83	86	87	89	87	90

L'approche n-grammes (colonne ngrm) étend la technique naïve en utilisant un contexte de *PdD* plutôt qu'un unigramme. Un syntagme est choisi en fonction des *PdD* qui l'encadrent. Le contexte est défini par une fenêtre centrée autour de la *PdD*. Les résultats rapportés ici sont obtenus en utilisant une fenêtre de taille 5, l'augmentation de la taille de la fenêtre n'améliorant pas les résultats (Johansson, 2000). La différence principale entre notre approche et la leur vient que (i) leur approche ne généralise pas les données et (ii) la technique de lissage utilisée ne garantit pas que les probabilités utilisées somment à 1.

Pour ce qui est de (i), il semble que même si nous généralisons, nous obtenons sensiblement les mêmes résultats. Cependant, il est important de noter que la généralisation nous permet d'obtenir des automates assez petits (environ 200 états et 8.000 transitions) qui sont sensiblement plus petits que le modèle n-grammes. L'aspect (ii) est moins important dans la mesure où le critère de qualité est indépendant du fait que le modèle soit consistant ou pas. On pourra cependant mentionner que l'obtention d'un modèle consistant autorise la combinaison du modèle avec d'autres approches statistiques.

Le modèle de Markov lexicalisé (LMM dans la table 1) prend les 479 mots les plus fréquents dans l'ensemble d'apprentissage (en excluant la ponctuation, les symboles les noms propres et les nombres) et divise leurs *PdD* correspondantes (Pla *et al.*, 2000). Cette lexicalisation permet une amélioration de l'ordre de 4% de la $F_{\beta=1}$. La différence principale entre leur approche et la notre vient du fait que la détection des *PdD* à diviser est chez nous automatique. De plus, notre méthode divise moins de *PdD* (100 comparativement à 470). Enfin, notre approche ne demande aucun traitement préliminaire des données. L'approche (Pla *et al.*, 2000) nécessite pour sa part la détection de noms propres qui n'est pas un problème résolu dans un cadre non supervisé par des méthodes automatique. Il faut donc faire un prétraitement des données manuel ou fournir un ensemble d'apprentissage spécifique pour résoudre ce problème. Enfin, l'argument de compacité de notre modèle demeure encore un point important ici.

9 Conclusions et perspectives

Nous avons présenté ici un algorithme pour faire de l'analyse syntaxique partielle via un algorithme d'inférence grammaticale ; nous avons construit dans un premier temps un automate pour modéliser une séquence de données, et nous l'avons utilisé comme un transducteur probabiliste. Les avantages de l'inférence grammaticale sont la compacité des modèles obtenus et le fait que nos modèles peuvent modéliser une dépendance de longueur non bornée. Il apparaît dans nos expérimentations, que nous pouvons obtenir une amélioration substantielle de nos résultats en utilisant une information lexicale restreinte (100 mots). Enfin, il semble que la technique du bagging puisse être utilisée avec succès pour améliorer la modélisation de densité de probabilités. A notre connaissance, c'est la première adaptation de la technique de bagging à l'estimation de densité de probabilités.

Dans nos travaux futurs, nous continuerons nos investigations sur la bagging, particulièrement, en examinant le comportement avec plus de B-ensembles et une étude de la variation de la taille des B-ensembles. Nous pensons également étudier le cas du boosting, que nous pensons adaptable à ce cadre.

Par ailleurs, les automates que nous inférons sont déterministes. Ceci implique, d'un certains point de vu, qu'ils ne peuvent pas utiliser le contexte futur, même si l'algorithme de Viterbi compense un peu ce fait. Nous aimerions étudier la possibilité d'inférer des modèles de séquences inverses et de combiner les prédictions des automates "forward" et "backward"[6].

---

6. Nous conservons ici la terminologie anglaise afin de faire le lien avec l'algorithme "forward-backward algorithm" qui permet l'estimation des probabilités dans un modèle de Markov caché.

RÉFÉRENCES

ABE N. & K.WARMUTH M. (1992). On the computational complexity of approximating distributions by probabilitic automata. *Machine Learning*, **9**, 205–260.

BREIMAN L. (1996). Bagging predictors. *Machine Learning*, **24**(2), 123–140.

BRILL E. (1995). Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics*, **21**(4), 543–565.

CARRASCO R. & ONCINA J. (1994). Learning stochastic regular grammars by means of a state merging method. In *Second Intl. Collo. on Grammatical Inference*, p. 139–152.

COLLINS M. (1997). Three generative, lexicalized models for statistical parsing. In P. R. COHEN & W. WAHLSTER, Eds., *Proc. of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics*, p. 16–23, Somerset, New Jersey: ACL.

COVER T. & THOMAS J. (1991). *Elements of Information Theory*. Wiley Interscience Publication.

GOODMAN J. (2001). *A bit of progress in Language Modeling*. Rapport interne, Microsoft Reserach.

HEDERSON C. J. & BRILL E. (2000). Bagging and boosting a treebank parser. In *NAACL*, p. 34–41, Seattle, Washington, USA.

JOHANSSON C. (2000). A context sensitive maximum likelihood approach to chunking. In *CoNLLL-2000 and LLL-2000*, p. 136–138, Lisbon, Portugal.

JURAFSKY D. & MARTIN J. H. (2000). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Englewood Cliffs, New Jersey: Prentice Hall.

LANG K., PEARLMUTTER B. & PRICE R. (1998). Results of the abbadingo one DFA learning competition and a new evidence-driven state merging algorithm. *Lecture Notes in Computer Science*, **1433**, 1–12.

MARCUS M., SANTORINI S. & MARCINKIEWICZ M. (1993). Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*, **19**(2), 313–330.

PLA F., MOLINA A. & PRIETO N. (2000). Improving chunking by means of lexical-contextual information in statistical language models. In *CoNLLL-2000 and LLL-2000*, Lisbon, Portugal.

RON D., SINGER Y. & TISHBY N. (1995). On the learnability and usage of acyclic probabilistic finite automata. In *ACM*, p. 31–40, Santa Cruz CA USA: COLT'95.

SANG E. T. K. (2000). Noun phrase recognition by system combination. In *Proceedings of BNAIC'00*, p. 335–336: Tilburg, The Netherlands.

SANG E. T. K. & BUCHHOLZ S. (2000). Introduction to the conlll-2000 shared task: Chunking. In *CoNLLL-2000 and LLL-2000*, p. 127–132, Lisbon, Portugal.

SCHÜTZE H. & SINGER Y. (1994). Part-of-speech tagging using a variable memory markov model. In *Meeting of the Association for Computational Linguistics*, p. 181–187.

STOLCKE A. & OMOHUNDRO S. Inducting probabilistic grammars by bayesian model merging. In SPRINGER, Ed., *ICGI'94*, p. 106–118.

THOLLARD F. (2001a). Improving probabilistic grammatical inference core algorithms with post-processing techniques. In *ICML'01*, p. 561–568, Williams: Morgan Kauffman.

THOLLARD F. (2001b). Inférence grammaticale probabiliste et détection de groupes nominaux : résultats préliminaires. In PUG, Ed., *CAp 2001*, p. 227–242, Grenoble.

VAN RIJSBERGEN C. J. (1975). *Information Retrieval*. London, England: Butterworths.