Modèles génératif et discriminant en analyse syntaxique : expériences sur le corpus arboré de Paris 7

Joseph Le Roux, Benoit Favre, Seyed Abolghasem Mirroshandel, Alexis Nasr

TALN 2011, Montpellier, 27 juin – 1er juillet 2011

Modèles génératif et discriminant en analyse syntaxique : expériences sur le corpus arboré de Paris 7

Joseph Le Roux Benoît Favre Seyed Abolghasem Mirroshandel Alexis Nasr
LIF - CNRS UMR 6166 - Université Aix Marseille
{joseph.le-roux, benoit.favre, alexis.nasr}@lif.univ-mrs.fr

Résumé. Nous présentons une architecture pour l'analyse syntaxique en deux étapes. Dans un premier temps un analyseur syntagmatique construit, pour chaque phrase, une liste d'analyses qui sont converties en arbres de dépendances. Ces arbres sont ensuite réévalués par un réordonnanceur discriminant. Cette méthode permet de prendre en compte des informations auxquelles l'analyseur n'a pas accès, en particulier des annotations fonctionnelles. Nous validons notre approche par une évaluation sur le corpus arboré de Paris 7. La seconde étape permet d'améliorer significativement la qualité des analyses retournées, quelle que soit la métrique utilisée.

Abstract. We present an architecture for parsing in two steps. First, a phrase-structure parser builds for each sentence an n-best list of analyses which are converted to dependency trees. Then these trees are rescored by a discriminative reranker. This method enables the incorporation of additional linguistic information, more precisely functional annotations. We test our approach on the French Treebank. The evaluation shows a significative improvement on different parse metrics.

Mots-clés : analyse syntaxique, corpus arboré, apprentissage automatique, réordonnancement discriminant.

Keywords: parsing, treebank, machine learning, discriminative reranking.

1 Introduction

On peut observer l'existence de deux approches en analyse syntaxique automatique. La première, dite générative, se fonde sur la tradition des langages formels et des systèmes de réécriture. L'analyse syntaxique est envisagée ici comme un processus permettant de passer d'une structure initiale (une chaîne d'entrée) à une structure finale (un arbre ou une forêt d'analyses). On utilise le plus couramment les grammaires algébriques qui peuvent s'analyser en temps polynomial. Malheureusement, l'hypothèse d'indépendance des réécritures qui sous-tend ce formalisme ne permet pas une analyse très fine de certains phénomènes, en particulier les dépendances à longue distance et les dépendances lexicales.

La seconde approche, dite discriminante, se fonde sur la « syntaxe comme théorie des modèles » (en anglais *model-theoretic syntax*, (Pullum & Scholz, 2001)) et a connu un regain d'intérêt grâce aux progrès réalisés dans le domaine de l'apprentissage automatique, plus précisément en classification automatique. Dans cette approche, la grammaire est vue comme un système de contraintes sur les structures syntaxiques correctes. Les mots de la phrase d'entrée sont eux-mêmes vus comme des contraintes sur les positions qu'ils occupent et l'analyse syntaxique revient à résoudre ces contraintes. Le problème majeur de cette seconde approche tient à sa complexité. Les contraintes pouvant en théorie porter sur divers aspects des structures finales, il n'est pas possible d'utiliser des techniques de programmation dynamique efficaces et il faut, dans le pire des cas, énumérer tous les arbres pour ensuite évaluer leur pertinence. Dans certains développements de cette approche, utilisés dans le présent travail, les contraintes sont uniquement d'ordre numérique. Une analyse y est représentée par un vecteur de traits et sa qualité se mesure par la distance entre ce dernier et l'analyse de référence.

Une manière de tirer profit des deux approches consiste, comme l'a proposé (Collins, 2000), à les combiner de façon séquentielle. Un analyseur de type génératif produit alors un ensemble de structures candidates pour un second module, discriminant, de façon à contraindre son espace de recherche. Cette approche par analyse puis réordonnancement (en anglais, *parsing/reranking*) est utilisée dans l'analyseur de Brown (Charniak & Johnson, 2005), adapté pour le français dans (Seddah *et al.*, 2009). Il est même intéressant de fournir au module de réordonnancement des analyses provenant de différents analyseurs, comme le montrent les résultats obtenus par (Johnson & Ural, 2010).

Notre architecture, représentée sur la figure 1, reprend ces deux étapes. Lors de la première étape, un analyseur syntagmatique traite chaque phrase d'entrée et produit la liste des n analyses les plus probables munies de leur probabilité. Elles sont ensuite annotées par un étiqueteur fonctionnel avec des fonctions syntaxiques classiques *sujet, objet, objet indirect...* Les analyses syntagmatiques enrichies d'annotations fonctionnelles sont alors converties en structures de dépendances. À l'issue de cette conversion, on dispose donc de candidats qui sont des structures de dépendances munies du score donné par le premier analyseur. Cette étape est réalisée par un analyseur syntagmatique PCFG-LA (Petrov *et al.*, 2006) couplé à l'étiqueteur et au convertisseur BONSAÏ [1].

Chaque structure de dépendances munie d'un score est ensuite traduite en un vecteur de traits. Ces traits représentent des configurations structurelles qui peuvent être absentes ou présentes dans l'arbre de dépendances et le vecteur indique leur nombre d'occurrences pour ce candidat. Le score attribué par l'analyseur syntagmatique est lui-même vu comme un trait. Les différents candidats sont finalement évalués par le réordonnanceur et le système retourne le meilleur candidat. Ce module est réalisé par notre implantation de l'algorithme MIRA (Crammer *et al.*, 2006).

Il nous paraît important de revenir ici sur deux aspects de notre architecture. Le premier est la réalisation de l'étape de réordonnancement sur des structures de dépendances et non pas sur des structures syntagmatiques, à l'image de (Charniak & Johnson, 2005). Notre analyseur produisant des structures syntagmatiques, il aurait en effet été plus naturel de réaliser le réordonnancement sur des structures syntagmatiques et de s'épargner ainsi leur conversion en structures de dépendances. Deux raisons sont à l'origine de ce choix. D'une part, il nous a semblé que de nombreuses contraintes se modélisent plus naturellement sous la forme de structures de dépendances que sous la forme de structures syntagmatiques. On pense en particulier à des contraintes sur les cadres de sous-catégorisation ou à des contraintes de sélection, qui font explicitement appel à la notion de fonction syntaxique. D'autre part, un certain nombre de travaux récents en analyse syntaxique (McDonald, 2006; Nivre *et al.*, 2007) reposent sur les structures de dépendances et il nous a semblé intéressant de proposer un système de réordonnancement pour ce type d'analyses.

1. disponibles sur le site `http://alpage.inria.fr/statgram/frdep/fr_stat_dep_parsing.html`.

La raison pour laquelle nous n'avons pas utilisé directement un analyseur en dépendances, qui aurait été sans doute un choix plus naturel, est qu'il n'existe pas à notre connaissance d'analyseur en dépendances qui génère les n analyses les plus probables. Les analyseurs de (McDonald, 2006), ou de (Nivre *et al.*, 2007), par exemple, ne permettent pas de les produire, même si le premier peut les approximer. C'est donc une raison pragmatique qui nous a poussés à faire ce choix. Enfin, il était intéressant de vérifier si, comme pour l'anglais, les structures de dépendances obtenues par conversions de structures syntagmatiques sont de meilleure qualité que celles renvoyées par les analyseurs en dépendances directement.

Le réordonnanceur proposé dans ce travail, qui sera décrit en détail dans la section 3, partage plusieurs de ses caractéristiques avec l'analyseur de (McDonald, 2006), évoqué ci-dessus. Les deux reposent sur l'algorithme d'apprentissage MIRA (Crammer *et al.*, 2006) décrit dans la section 3. De plus, les contraintes considérées dans notre modèle sont inspirées de (McDonald, 2006). La différence fondamentale provient du fait que les seules analyses prises en comptes sont celles produites par le modèle génératif (il s'agit d'un réordonnanceur et non d'un analyseur). L'avantage de cette solution par rapport à (McDonald, 2006) est que nous ne sommes pas restreints à un ensemble de traits locaux. En effet la prise en compte de traits non prévus, de domaine de localité arbitraire, dans l'algorithme de (McDonald, 2006) suppose des modifications de l'algorithme d'analyse alors qu'ils peuvent être ajoutés facilement dans notre modèle de réordonnancement.

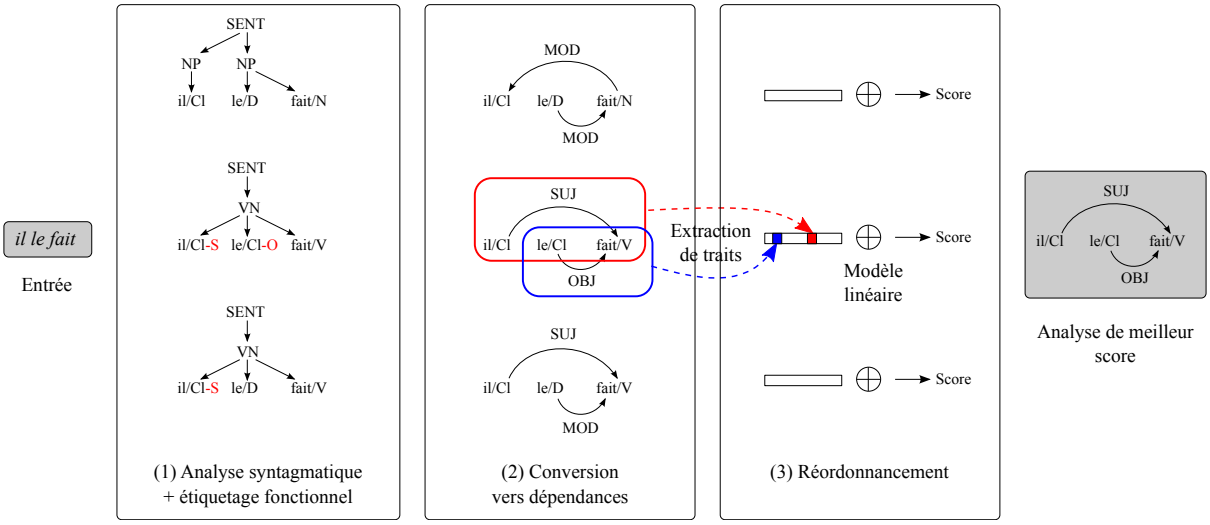

FIGURE 1 – Architecture de notre analyseur : (1) génération de n arbres syntagmatiques annotés en fonctions, (2) conversion vers une représentation en dépendances et extraction de vecteurs de traits, (3) calcul des scores à l'aide d'un modèle linéaire. L'analyse de meilleur score est considérée comme l'analyse finale.

La suite de l'article se présente de la façon suivante : nous décrivons en 2 les détails du modèle utilisé dans notre analyseur génératif puis en 3 le modèle de réordonnancement discriminant et les patrons de traits utilisés. La section 4 présente les résultats obtenus sur le corpus arboré développé à l'université Paris 7 (Abeillé *et al.*, 2003) et la section 5 présente les conclusions.

2 Modèle génératif

La première partie de notre système, l'analyse syntaxique classique, produit des structures en dépendances de surface grâce à un système séquentiel, à l'image de (Candito *et al.*, 2009, 2010b). Un analyseur, fondé sur les PCFG-LA, produit des structures syntagmatiques qui sont ensuite transformées en arbres de dépendances. Deux points nous distinguent des travaux précédents : (1) la liste d'analyses candidates est produite par un nouvel analyseur et (2) ces analyses ne sont pas considérées comme les structures finales mais seront traitées dans le réordonnanceur.

2.1 Les grammaires algébriques à annotations latentes

Les grammaires algébriques probabilistes à annotations latentes (PCFG-LA), introduites par (Matsuzaki *et al.*, 2005), peuvent être vues comme une façon de spécialiser automatiquement le jeu d'étiquettes d'une grammaire algébrique (PCFG) à partir d'un corpus de manière à en améliorer la précision.

Chaque symbole de la grammaire est enrichi d'annotations se comportant comme des sous-classes de ce symbole, et les probabilités des règles qui manipulent les symboles augmentés sont estimées par la méthode EM d'apprentissage non-supervisé, à partir des fréquences relatives observées sur le corpus. (Petrov *et al.*, 2006) proposent d'apprendre ces grammaires en plusieurs rondes : à chaque itération on divise une annotation d'un symbole en deux si l'apport des nouvelles annotations augmente la vraisemblance du corpus d'entraînement. Cette méthode permet d'obtenir une grammaire dans laquelle le nombre d'annotations est adapté au symbole et beaucoup plus compacte que celles obtenues par (Matsuzaki *et al.*, 2005).

On peut reprendre l'illustration des spécialisations d'étiquettes de parties du discours de (Petrov *et al.*, 2006) pour le français. Par exemple l'étiquette DET est divisée en quinze sous-étiquettes après cinq rondes d'apprentissage. Nous montrons dans la table 1 les trois mots les plus fréquents pour chaque annotation [2]. Même si la spécialisation est difficile à interpréter complètement, on note que les articles définis et indéfinis sont séparés des démonstratifs et possessifs d'une part et des cardinaux d'autre part. Il est important de noter que cette distinction est apprise par une méthode qui dépend largement des paramètres d'initialisation et qui ne garantit pas de trouver la spécialisation optimale.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	une	la	l'	un	les	les	ses	le	cette	son	NB	deux	NB	NB	NB
2	la	La	la	son	des	Les	ces	Le	Cette	ce	10	trois	7	40	1
3	La	L'	les	Un	Les	de	leurs	un	Ce	leur	3	quelques	trois	20	30

TABLE 1 – Division du symbole DET en annotations

Dans ce formalisme, le problèmes de l'analyse exacte devient NP-difficile mais (Petrov & Klein, 2007) décrivent comment l'approximer efficacement, en tirant également profit de la structure hiérarchique des annotations créées lors des rondes successives. Une forêt d'analyses est produite avec la PCFG d'origine et est ensuite élaguée avec les grammaires intermédiaires produites lors des rondes successives d'apprentissage [3], c'est-à-dire qu'on ne garde qu'un sous-ensemble *a priori* intéressant des analyses, de manière à restreindre au maximum l'espace de recherche lors du décodage avec la grammaire finale.

Nous utilisons notre propre analyseur de PCFG-LA [4]. La grammaire est apprise en cinq rondes sur le corpus d'entraînement : c'est le nombre de rondes optimal sur notre corpus. Pour pouvoir traiter les mots inconnus, les mots rares sont remplacés par des chaînes contenant des informations positionnelles et typographiques, en particulier leur suffixe. La liste des suffixes *intéressants* est collectée sur ce corpus en fonction du gain d'information pour l'étiquetage et permet de mieux traiter les mots inconnus (Attia *et al.*, 2010).

Ce type de grammaire a déjà été utilisé pour le français à partir du même corpus (Crabbé & Candito, 2008). Il a donné des résultats du niveau de l'état de l'art en matière d'analyse en constituants. (Candito & Crabbé, 2009) montrent que si l'on sait catégoriser les mots en classes lexicales (agrégats ou *clusters*), ces grammaires offrent les meilleures analyses en syntagmes pour le français. Nous reprendrons cette classification pour nos expériences (cf. section 4). Cependant, les structures en dépendances extraites des meilleures analyses en constituants ne sont pas aussi bonnes que celles obtenues directement par un analyseur en dépendances (Candito *et al.*, 2010b).

2.2 Structures de dépendances

Bien qu'une théorie syntaxique puisse se représenter à partir de syntagmes et de dépendances (Rambow, 2010), certains types d'informations sont plus ou moins faciles à décrire selon la représentation choisie. L'analyseur génératif étant appris sur des structures syntagmatiques, il est vraisemblable qu'une partie de l'information linguistique lui échappe parce qu'elle est implicite ou difficile à retrouver dans cette représentation.

2. Le symbole terminal NB est une chaîne générique qui remplace les nombres qui apparaissent peu de fois dans le corpus d'entraînement.
3. En réalité ces grammaires sont recalculées à la volée à partir de la grammaire finale et de la structure hiérarchique des annotations.
4. Cet analyseur est disponible pour les travaux académiques. Les lecteurs intéressés peuvent contacter le premier auteur.

Cette équivalence n'est de toute façon vraie que si les deux représentations offrent réellement les mêmes informations. Or, comme cela sera décrit dans la section 4, nous avons fait disparaître les fonctions du corpus d'apprentissage de l'analyseur syntagmatique pour aider lutter à contre l'effet de dispersion des données et améliorer l'apprentissage des relations de sous-constituance. Mais pour obtenir les relations de dépendances typées, il est nécessaire de disposer de l'information fonctionnelle.

Pour passer des structures en constituants aux structures en dépendances, nous utilisons le convertisseur développé par (Candito *et al.*, 2010a) et disponible dans la boîte à outils BONSAÏ. La conversion se décompose en deux étapes :

1. Les nœuds internes sont réannotés avec des étiquettes fonctionnelles, en utilisant un classifieur multiclasse par maximum d'entropie.
2. Les arbres ainsi décorés sont convertis en structures de dépendances par un ensemble de règles de propagations de têtes fonctionnelles et d'heuristiques.

Cette conversion effectuée, les analyses sont prêtes à être réordonnancées.

3 Modèle discriminant

Le modèle discriminant que nous proposons repose sur l'algorithme Margin-infused Relaxed Algorithm (MIRA) (Crammer *et al.*, 2006). Selon ce modèle, le score d'une analyse est calculé comme la combinaison linéaire des traits extraits à partir de cette analyse, pondérés par un vecteur de poids représentant les paramètres du modèle. MIRA, l'algorithme d'apprentissage des paramètres du modèle, est très similaire au Perceptron (Rosenblatt, 1958), et est donc rapide et peu gourmand en ressources, tout en offrant de meilleures performances.

Le réordonnancement discriminant des analyses se déroule selon deux étapes : apprentissage des paramètres du modèle et prédiction. L'étape de prédiction consiste à produire les n meilleures analyses du modèle génératif, extraire des traits pour caractériser ces analyses et attribuer un score à chaque analyse en fonction de ses traits et du modèle discriminant. L'analyse de meilleur score est alors sélectionnée comme sortie finale du système. L'étape d'entraînement consiste à utiliser des exemples de phrases avec leurs analyses de référence (ensemble d'apprentissage) pour déterminer les paramètres du modèle. Alors que la plupart des modèles discriminants tentent de minimiser le taux d'erreur global sur l'ensemble des exemples d'apprentissage, MIRA se contente de traiter les exemples un par un, ajustant son modèle pour que l'analyse sélectionnée pour la phrase courante soit celle qui est la plus proche de l'analyse de référence. Une telle approche, appelée « en ligne », limite les ressources nécessaires, processeur et mémoire, rendant possible l'apprentissage de modèles qui prennent en compte un très grand nombre de traits.

3.1 Définitions

On se place dans un espace vectoriel de dimension m où chaque dimension correspond à un trait. Certaines dimensions représentent la présence ou l'absence d'un trait (valeur booléenne), son nombre d'occurrence dans l'analyse (entier naturel), ou une valeur réelle quelconque (par exemple la probabilité de l'analyse, ou son logarithme, selon le modèle génératif). Une analyse p est alors représentée sous la forme d'un vecteur de réels $\phi(p)$. Dans le cas d'un indicateur de présence, la i^{e} coordonnée de $\phi(p)$ vaut 1 si p possède le i^{e} trait et 0 sinon. Un tel vecteur est généralement creux (une analyse ne possède en moyenne qu'une petite partie des différents traits possibles). Un modèle est un vecteur de poids w de dimension m dont la i^{e} coordonnée est le poids associé au i^{e} trait. Plus ce poids est important, plus le trait correspondant aura été jugé discriminant. Le score d'une analyse p n'est rien d'autre que le produit scalaire du vecteur $\phi(p)$ et du vecteur w :

$$score(p) = \sum_{i=1}^{m} w_i \times \phi_i(p) \tag{1}$$

Soit L la liste des n meilleures analyses produites par l'analyseur syntaxique génératif pour une phrase. On calcule le score de chaque analyse et l'analyse de score maximum $\hat{p}$ est choisie comme sortie finale du système :

$$\hat{p} = \underset{p \in L}{\text{argmax}}\, score(p) \tag{2}$$

La phase d'apprentissage consiste à utiliser les phrases d'entraînement et leurs analyses de référence pour déterminer le vecteur de poids w. Le classifieur MIRA commence avec un vecteur de poids nul (c'est-à-dire que toutes les analyses ont un score de zéro), et essaie de modifier ce vecteur de façon à ce que les bonnes analyses ait un score plus élevé que les mauvaises analyses. L'analyse la plus proche de la référence est nommée *oracle* (notée o). Il serait souhaîtable que l'oracle ait le meilleur score parmi les analyses proposée pour une phrase. Soit $error(p)$ le nombre de mauvaises dépendences (étiquette, position, direction) dans l'analyse p. L'oracle o est l'analyse pour laquelle $erreur()$ est minimale.

Les phrases de l'ensemble d'entraînement sont traitées séquentiellement. Pour chacune d'entre elles, on détermine la liste des n meilleures analyses candidates, puis l'analyse candidate de meilleur score, notée $\hat{p}$. Si cette analyse est différente de l'oracle ($\hat{p} \neq o$), cela signifie que le vecteur de poids w peut être amélioré. Dans ce cas, on recherche une modification de w qui assure que o ait un score plus élevé que l'analyse qui avait le meilleur score. Plus précisément, on souhaite que la différence entre leurs scores soit proportionnelle à la différence entre leur distance à la référence. Ainsi, une très mauvaise analyse aura un score bien plus faible qu'une analyse de qualité moyenne. Trouver une amélioration de w peut être formulé comme un problème d'optimisation sous la contrainte que la différence des scores de l'oracle et de l'hypothèse de plus haut score soit supérieure à la différence entre leurs distances à la référence. Comme il existe une infinité de vecteurs w satisfaisant cette contrainte, on recherche celui de plus petite norme. Ce problème s'écrit sous la forme suivante :

$$\text{minimiser : } ||w|| \text{ tel que : } score(o) - score(\hat{p}) \geq error(o) - error(\hat{p}) \tag{3}$$

Des méthodes classiques d'optimisation quadratique sous contrainte sont utilisées : tout d'abord la contrainte est introduite dans la fonction objective grâce à des multiplicateurs de Lagrange. De cette façon, les solutions qui violent le plus la contrainte sont pénalisées par rapport aux autres solutions. Enfin, la méthode de Hildreth donne la solution analytique suivante au problème quadratique non contraint :

$$w^\star = w + \alpha\left[\phi(o) - \phi(\hat{p})\right] \tag{4}$$

$$\alpha = max\left[0, \frac{error(o) - error(\hat{p}) - (score(o) - score(\hat{p}))}{||\phi(o) - \phi(\hat{p})||^2}\right] \tag{5}$$

Ici, $w^\star$ est le nouveau vecteur de poids, α est un taux de modification et $[\phi(o) - \phi(\hat{p})]$ est la différence entre le vecteur de traits de l'oracle et celui de l'analyse de plus haut score. Concrètement, cette mise à jour attire le vecteur de poids en direction de l'oracle et l'éloigne de $\hat{p}$. Cet algorithme d'apprentissage est très proche de l'algorithme du perceptron, et tout comme pour le perceptron, il est recommandé (1) de faire de multiples passes sur l'ensemble d'apprentissage et (2) de sauvegarder le vecteur de poids après chaque mise à jour et d'en faire la moyenne pour produire le vecteur de poids final [5]. L'algorithme 1 présente l'apprentissage MIRA.

3.2 Traits utilisés

La qualité d'un réordonnanceur dépend de la capacité de l'algorithme d'apprentissage à attribuer un bon poids aux traits discriminants, mais aussi, de manière cruciale, à la qualité des traits qui lui sont fournis. Ces traits peuvent porter sur n'importe quelle configuration lexico-syntaxique d'une analyse. L'ensemble des traits potentiels est par conséquent gigantesque. Pour être pertinent, un trait doit d'une part être assez général pour apparaître souvent et, d'autre part, permettre de discriminer les bonnes analyses des mauvaises. Il n'existe pas de méthode générale de sélection des traits pertinents, du fait de leur nombre et de leur caractère non monotone : un trait simple peut se révéler non pertinent mais l'extension de ce trait simple en un trait plus complexe peut l'être. L'espace des traits est par conséquent difficile à explorer systématiquement et la sélection des traits pertinents est une activité qui relève de l'art, pour reprendre un bon mot de (Charniak & Johnson, 2005).

Nous nous sommes inspirés de traits utilisés dans l'analyseur (McDonald, 2006) qui a montré de bonnes performances dans de nombreuses langues. Ils se divisent en cinq familles, chaque famille correspondant à un type de configuration. L'instanciation de ces patrons sur les sorties de l'analyseur a généré plus de 70 millions de traits. Nous décrivons ci-dessous les cinq familles de traits que nous illustrons sur la phrase *Les enfants mangent des glaces avec appétit* dans laquelle on s'intéressera en particulier à la dépendance objet *(mangent, glaces)*.

5. Dans la pratique, il n'est pas nécessaire de sauvegarder tous les vecteurs de poids, mais seulement deux vecteurs.

Algorithme 1 Entraînement MIRA

pour $i = 1$ à t **faire**
  pour chaque phrase de l'ensemble d'entraînement **faire**
    Générer les n meilleures hypothèses de l'analyseur en constituants.
    pour chaque hypothèse **faire**
      Extraire un vecteur de traits à partir de l'hypothèse.
      Calculer son score comme le produit scalaire entre ce vecteur et le vecteur de poids (éq. 1)
    fin pour
    Soit l'oracle, l'hypothèse la plus proche de la référence pour cette phrase.
    si l'hypothèse de meilleur score n'est pas l'oracle **alors**
      Calculer la différence entre le vecteur de traits de l'oracle et le vecteur de traits de l'hypothèse.
      Calculer le facteur α qui assure que l'oracle ait un meilleur score la prochaine fois (éq. 5)
      Ajouter au vecteur de poids ce facteur fois la différence entre les deux vecteurs (éq. 4)
    fin si
  fin pour
fin pour
Retourner un vecteur de poids moyen consituté à partir de l'état du vecteur de poids après chaque phrase d'entraînement.

Unigramme Les traits unigrammes sont les plus simples, ils ne mettent en jeu qu'une seule dépendance. Étant donné une dépendance entre les positions i et j de type l, gouvernée par x_i, notée $x_i \xrightarrow{l} x_j$, on crée deux traits, l'un pour le gouverneur x_i , l'autre pour le dépendant x_j qui prennent la forme de sextuplets (mot, lemme, partie du discours, statut dans la dépendance (gouverneur (G) ou dépendant (D)) , direction de la dépendance (droite (D) ou gauche (G)), type de la dépendance). On ajoute aussi tous les tuples avec une partie de l'information masquée pour lutter contre la dispersion des données lors de l'apprentissage.

Ainsi, la présence de la dépendance objet dans notre exemple donnera naissance aux deux traits :

`[mangent, manger, V, G, D, objet]` et `[glaces, glace, N, D, D, objet]`

mais aussi à tous les traits que l'on peut construire à partir des deux premiers par sous-spécification :

`[-, manger, V, G, D, objet]`, `[mangent, -, V, G, D, objet]` ...
`[mangent, -,-,-,-,objet]`

Ce processus de sous-spécification des traits s'applique à toutes les familles de traits, on omettra de le répéter ci-dessous.

Bigramme Contrairement à la famille précédente qui ne prenait en compte qu'un des deux membres d'une dépendance, les traits bigrammes modélisent la cooccurrence des deux membres de la dépendance, à l'image des dépendances bi-lexicales de (Collins, 1997). Étant donné la dépendance $x_i \xrightarrow{l} x_j$, on crée un trait (mot x_i, lemme x_i, partie du discours x_i, mot x_j, lemme x_j, partie du discours x_j, distance [6] de i à j, direction de la dépendance, type de la dépendance).

L'exemple ci-dessus donnera donc naissance au trait suivant :

`[mangent, manger, V, glaces, glace, N, 2, D, objet]`

où 2 est la distance séparant *mangent* et *glaces* dans la chaîne linéaire.

Contexte linéaire Contrairement aux deux familles précédentes qui s'intéressaient à une dépendance indépendamment de sa réalisation dans la chaîne, on regarde ici les éléments qui séparent un gouverneur de son dépendant dans cette dernière. Étant donné la dépendance $x_i \xrightarrow{l} x_j$ On crée un trait avec les partie de discours de x_i, de x_j et de chaque mot entre les positions i et j. On crée également un trait comportant les parties de discours aux positions $i-1$, i, $i+1$, $j-1$, j, $j+1$. La dépendance objet de notre exemple donnera naissance aux deux traits :

`[V, D, N]` et `[N, V, D, N, P]`

Contexte syntaxique, nœuds frères Cette famille de traits ainsi que la suivante mettent en jeu deux dépendances dans deux configurations particulières. Étant donné deux dépendances $x_i \xrightarrow{l} x_j$ et $x_i \xrightarrow{m} x_k$, on crée un trait (mot x_i, lemme x_i, partie du discours x_i, mot x_j, lemme x_j, partie du discours x_j, mot x_k, lemme x_k, partie du discours x_k, distance de i à j, distance de i à k, direction de la première dépendance,

6. Cette distance est réduite à sept classes selon qu'elle est égale à 1, 2, 3, 4, 5, comprise entre 5 et 10, ou supérieure à 10.

type de la première dépendance, direction de la seconde dépendance, type de la seconde dépendance). Ce qui donne, dans notre exemple : `[mangent, manger, V, glaces, glace, avec, avec, P, 2, 3, D, objet, D, mod]`

Contexte syntaxique, chaînes Étant donné deux dépendances $x_i \xrightarrow{l} x_j \xrightarrow{m} x_k$, on crée un trait (mot x_i, lemme x_i, partie du discours x_i, mot x_j, lemme x_j, partie du discours x_j, mot x_k, lemme x_k, partie du discours x_k, distance de i à j, distance de i à k, direction de la première dépendance, type de la première dépendance, direction de la seconde dépendance, type de la seconde dépendance). Dans notre exemple : `[mangent, manger, V, avec, avec, P, appétit, appétit, N, 3, 4, D, mod, D, objet]`

On peut noter que les patrons de traits ne reposent que sur des connaissances présentes dans les données d'apprentissage. Nous n'avons pas ajouté de traits qui proviennent de connaissances linguistiques externes.

4 Expériences

Dans cette section, nous évaluons les performances de notre analyseur. Nous présentons d'abord le module génératif seul, puis les deux modules ensemble.

Nous utilisons dans nos expériences le corpus arboré de Paris 7 (Abeillé *et al.*, 2003) (dans la suite, FTB). Il contient 12 350 phrases annotées syntaxiquement en constituants, et en étiquettes fonctionnelles. Ce n'est pas directement ce corpus que nous manipulons mais deux transformations de celui-ci.

1. La première, FTB-UC, mise au point par (Crabbé & Candito, 2008), garde la structure en constituants mais simplifie le jeu d'étiquettes. En particulier, les fonctions disparaissent et les informations morphologiques sont réduites. C'est sur ce corpus que nous apprendrons la grammaire syntagmatique.
2. La seconde, FTB-UC-DEP, présentée dans (Candito *et al.*, 2009), est une conversion du FTB en dépendances. Cette conversion est réalisée à l'aide de règles de propagation de têtes et d'heuristiques. C'est sur ce second corpus que l'on apprendra l'analyseur discriminant.

Pour entraîner et évaluer notre système, nous divisons ces corpus en 3 : une partie pour l'entraînement (80%), une partie pour le développement (10%) et le reste pour l'évaluation finale.

Nous employons deux types de grammaires syntagmatiques, le premier appris directement sur FTB-UC (modèle simple), le second appris sur une version modifiée de FTB-UC dans laquelle les mots sont remplacés par leur classe d'équivalence (modèle agrégats), comme dans (Candito & Crabbé, 2009).

4.1 Évaluation du modèle génératif seul

Les performances de notre analyseur syntagmatique sur le corpus de développement sont résumées dans le tableau 2. Le F-score [7] est la moyenne harmonique du rappel (ici, les constituants de la référence retrouvés par l'analyseur) et de la précision (ici, les constituants prédits présents dans la référence). Nous donnons les scores oracles de notre analyseur quand il renvoie les 1, 10, 50 et 100 analyses les plus probables d'une phrase, pour donner une idée de la marge de progression possible.

Les résultats quantitatifs de la conversion en structures de dépendances sont également présentés dans la table 2. Le score d'attachement étiqueté (LAS) est le taux de dépendances typées correctement reconnues [8] par l'analyseur. Le score non-étiqueté (UAS) est ce même taux lorsque l'on ne tient pas compte du type des dépendances. Nous avons représenté sur la dernière colonne (GOLD) l'évaluation de la conversion en dépendances de l'analyse syntagmatique de référence. Ces résultats nous permettent d'évaluer la qualité de l'étiquetage fonctionnel, ils montrent que l'étiqueteur effectue à peu près 4% d'erreurs dans l'attribution de ces étiquettes. Comme précédemment, nous donnons aussi le score oracle quand notre analyseur renvoie plusieurs analyses.

7. Nous utilisons le logiciel `evalb` que nous avons modifié pour qu'il donne également le score oracle quand l'analyseur fournit une liste d'arbres.
8. La ponctuation n'est pas prise en compte.

	DEV-1	DEV-10	DEV-50	DEV-100	GOLD
F	84,41	89,35	91,71	92,56	100
LAS	86,01	89,25	90,86	91,48	96,04
UAS	89,60	92,70	94,23	94,80	100
F agrégats	85,02	89,98	92,33	93,27	100
LAS agrégats	86,99	89,93	91,61	92,25	96,04
UAS agrégats	90,72	93,48	95,05	95,68	100

TABLE 2 – Scores de l'analyseur génératif sur la partie de développement

Les résultats donnés ci-dessus nous permettent de tirer deux conclusions importantes. D'une part les résultats de l'analyseur sont du niveau de l'état de l'art pour l'analyse syntagmatique du français ($84,41\%$ de F-score). D'autre part, la marge de progression du réordonnanceur est importante puisque le score oracle LAS sur les 100 analyses les plus probables est de $91,48\%$ alors que le score de l'analyse la plus probable est de $86,01\%$ soit une marge possible de progression de $5,47\%$.

4.2 Ajout du réordonnanceur

4.2.1 Apprentissage

Le modèle discriminant, c'est-à-dire les instances des patrons de traits et leur poids, est appris sur le corpus d'entraînement. L'analyseur génératif produit 100 analyses par phrase [9] pour ce corpus qui servent d'exemples d'apprentissage au réordonnanceur. Le modèle donne 71 millions de traits pour la grammaire simple et 75 millions pour la grammaire d'agrégats. Notez qu'un tel nombre de traits n'est pas pénalisant car l'algorhtme discriminant ne donne un poids non nul qu'aux traits utiles.

4.2.2 Évaluation

	base	10	20	50	100
F	84,41	85,38	**85,58**	85,55	85,32
LAS	86,01	87,06	87,31	**87,39**	87,28
UAS	89,60	90,57	90,77	**90,83**	90,69

grammaire simple

	base	10	20	50	100
F	85,00	85,94	86,12	**86,20**	86,15
LAS	86,99	88,00	88,06	88,10	**88,17**
UAS	90,78	91,54	91,57	91,58	**91,62**

grammaire apprise sur agrégats

TABLE 3 – scores du réordonnanceur en fonction du nombre de candidats

Pour notre évaluation, nous avons testé plusieurs configurations sur le corpus de développement en faisant varier le nombre de candidats fourni au réordonnanceur lors de la phase de prédiction [10]. Les résultats sont présentés dans la table 3. Pour la métrique LAS, les meilleurs résultats sont obtenus en donnant 50 candidats au réordonnanceur dans le cas de la grammaire simple et 100 dans le cas de la grammaire d'agrégats. Mais la différence avec les autres configurations n'est pas significative.

Puisque la meilleure configuration pour les différentes grammaires n'est pas la même selon la métrique utilisée mais que la configuration à 50 candidats est toujours la meilleure selon l'une d'entre elles, c'est cette configuration qui est utilisée sur le corpus de test pour l'évaluation finale avec la grammaire simple et la grammaire d'agrégats. Les résultats, dans la table 4, sont du même ordre de grandeur [11]. Pour la grammaire simple, le réordonnanceur de notre système permet de passer d'un F-score de 85,09% à 86,02%, pour le LAS de 86,68% à 87,91% et pour le UAS de 90,22% à 91,31%. Sur les 3 métriques le réordonnanceur montre une amélioration significative [12]. Il est intéressant de noter que nos traits portent uniquement sur les structures en dépendances (mis à part le score

9. Pour certaines phrases, les phrases courtes en particulier, notre système renvoie moins de 100 analyses.
10. Les poids des traits sont toujours appris avec 100 candidats par phrase.
11. $F < 40$ est le F-score en constituants pour les phrases de moins de 40 mots.
12. Les différences de scores entre le système de base et les nouveaux systèmes incorporant le module discriminant sont statistiquement significatives avec une valeur $p < 0.01$. Les différences entre les nouveaux systèmes ne le sont pas.

attribué par l'analyseur PCFG-LA) et que le F-score qui mesure la qualité des arbres syntagmatiques est tout de même améliorée.

	base	réord.
F	85,09	86,02
F < 40	87,10	87,82
LAS	86,68	87,91
UAS	90,22	91,31

grammaire simple

	base	réord.
F	86,35	86,89
F < 40	88,45	88,74
LAS	87,37	88,45
UAS	91,08	91,91

grammaire apprise sur agrégats

TABLE 4 – Scores du système sur le corpus de test

4.2.3 Comparaisons

	F < 40	LAS	UAS
Ce travail	87,82	87,91	91,31
Ce travail + agrégats	**88,74**	**88,45**	**91,91**
MATE + MELT	–	88,17	90,15
BKY	88,2	86,8	91,0
MST	–	88,2	90,9

TABLE 5 – Comparaisons des différents résultats d'analyse syntaxique

Nous donnons un tableau récapitulatif de différents résultats d'analyseurs sur le FTB dans la table 5. Nous comparons notre système (nous avons choisi les configurations qui donnaient les meilleurs scores LAS sur le corpus de développement) avec l'analyseur en dépendances MATE (Bohnet, 2010), entraîné et évalué avec MELT (Denis & Sagot, 2010) pour l'étiquetage en parties du discours. Nous citons aussi les travaux de comparaisons de (Candito *et al.*, 2010b), avec un premier système (BKY dans la table) proche du nôtre avec un analyseur syntagmatique qui fournit une sortie convertie en arbre de dépendances. Le second système (MST dans la table) est l'analyseur MSTParser de (McDonald *et al.*, 2005). Ces deux systèmes utilisent aussi les agrégats et non directement les mots du texte.

4.2.4 Analyse

Une analyse du vecteur de poids produit par le modèle discriminant montre que seuls 27% des 75 millions de traits observés dans les données d'entraînement correspondent à des poids non nuls (modèle avec agrégats). Les autres traits ont donc été jugés non discriminants. Nous avons analysé les 1000 traits de plus grand poids positif, représentant les caractéristiques jugées les plus pertinentes pour discriminer les bonnes analyses et les 1000 traits de poids négatifs de plus grande valeur absolue, symptomatiques des mauvaises analyses.

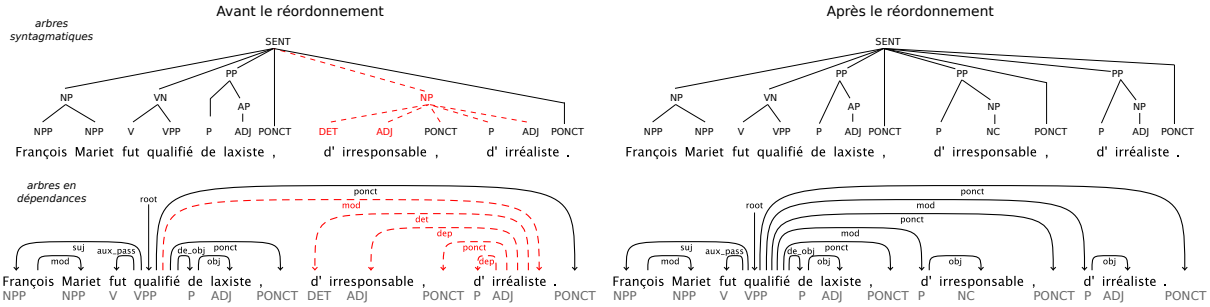

FIGURE 2 – Exemple de réordonnancement bénéfique (163e phrase de développement) : la meilleure analyse selon le modèle génératif est à gauche, la meilleure selon le modèle discriminant, la 35e hypothèse du modèle génératif, est à droite. Les erreurs, indiquées par des pointillés, sont propagées lors de la conversion en dépendances.

Cette analyse fait apparaître que les traits issus du repli [13] sont utiles pour caractériser de mauvaises analyses (63% des 1000 poids les plus négatifs). De plus, les traits négatifs font souvent référence à un indicateur d'échec de la lemmatisation. Comme cette lemmatisation est produite à partir d'un lexique et de la paire forme / partie du discours, et sous l'hypothèse que notre lexique est suffisamment riche, une erreur signifie que l'étiquetage en partie du discours est mauvais.

Le trait positif le plus discriminant est, sans surprise, le score donné par l'analyseur. Les autres traits positifs importants permettent d'avantager des étiquetages de parties du discours en fonction du contexte linéaire. On remarque aussi l'existence de traits qui incitent les conjoints à avoir la même catégorie dans les coordinations.

On voit donc que le réordonnancement peut remettre en question l'étiquetage de l'analyseur mais aussi qu'il peut influencer le traitement de phénomènes plus complexes comme la coordination. La figure 2 donne un exemple de phrase dont la meilleure analyse a été corrigée par le modèle discriminant.

Les patrons les plus représentés en positif sont **unigramme**, **bigramme**, **contexte linéaire**, **chaînes**, et leurs versions relachées, et en négatif **unigramme**, **bigramme**, **chaînes**, **nœuds frères**. Une analyse des résultats par type de dépendance montre que le réordonnanceur fait moins d'erreurs uniformément sur l'ensemble des types. La figure 2 donne un exemple de phrase dont l'analyse a été corrigée par le modèle discriminant.

5 Conclusion

Nous avons montré que l'ajout d'un module discriminant permet d'améliorer la qualité des analyses, comme nous avons pu le vérifier expérimentalement sur le corpus FTB à l'aide de trois métriques (F-score Parseval, LAS, UAS). Cependant, le gain est moins important que celui observé pour l'anglais (Charniak & Johnson, 2005). Sans procéder à une analyse d'erreurs exhaustive, on peut toutefois évoquer plusieurs points pouvant être améliorés.

En premier lieu, l'approche séquentielle est vulnérable aux erreurs en cascade. Bien que l'analyseur génératif fournisse plusieurs candidats, ce n'est pas le cas de l'étiqueteur fonctionnel. Les erreurs d'étiquetage ne sont donc pas récupérables. On peut envisager deux solutions ici : (1) permettre à l'étiqueteur de renvoyer une sortie ambiguë et laisser le réordonnanceur décider du meilleur étiquetage, et (2) utiliser des techniques plus sophistiquées dans la phase discriminante comme la prédiction structurée. On pourra ainsi se passer de l'étiqueteur.

Ensuite, cette approche à deux niveaux ne permet pas d'atteindre l'oracle, ce qui a déjà été observé sur l'anglais. Il est difficile de trouver un jeu de traits suffisamment général et qui soit utile pour une phrase particulière. Il y a encore des investigations à mener pour trouver un jeu de traits optimal, par exemple sur l'utilisation plus systématique de traits sur les structures de dépendances (par exemple en passant par des noyaux d'arbres), ou celle de connaissances externes, linguistiques ou collectées sur corpus (préférences lexicales, cadres de sous-catégorisation, verbes copulatifs, symétrie de la coordination...).

Enfin, notre système doit encore être amélioré pour une utilisation en situation réelle (par exemple, l'analyse de gros volumes de données provenant de la toile). L'extraction des traits pour le réordonnanceur, et vraisemblablement pour l'étiqueteur fonctionnel, est clairement le goulet d'étranglement. À titre d'exemple, pour analyser les 1235 phrases du corpus de test, les temps d'analyse en constituants, d'étiquetage fonctionnel et de conversion en structures de dépendances, d'extraction de traits et finalement de réordonnancement, sont respectivement : 9 min 55 s, 25 min 43 s, 51 min 12 s et 4 min 03 s.

Ce travail ouvre la voie à l'utilisation de données non étiquetées en plus d'un corpus arboré pour apprendre une meilleure grammaire générative (*self training* comme (McClosky *et al.*, 2008)) où le réordonnanceur évite au système d'apprendre sur les erreurs commises par l'analyseur génératif.

Remerciements

Ces travaux sont en partie financés par le projet ANR Sequoia ANR-08-EMER-013. Nous tenons à remercier Marie Candito qui nous a aidés à maîtriser BONSAÏ, Djamé Seddah qui nous a suggéré de tester notre architecture sur les grammaires d'agrégats, ainsi que les relecteurs anonymes.

13. C'est-à-dire qu'une partie de l'information est masquée (cf. section 3.2).

Références

ABEILLÉ A., CLÉMENT L. & FRANÇOIS T. (2003). *Treebanks*, chapter Building a treebank for French. Kluwer, Dordrecht.

ATTIA M., FOSTER J., HOGAN D., LE ROUX J., TOUNSI L. & VAN GENABITH J. (2010). Handling Unknown Words in Statistical Latent-Variable Parsing Models for Arabic, English and French. In *Proceedings of SPMRL*.

BOHNET B. (2010). Top Accuracy and Fast Dependency Parsing is not a Contradiction. In *Proceedings of COLING*.

CANDITO M., CRABBÉ B. & DENIS P. (2010a). Statistical French Dependency Parsing : Treebank Conversion and First Results. In *Proceedings of LREC2010*.

CANDITO M.-H. & CRABBÉ B. (2009). Improving Generative Statistical Parsing with Semi-Supervised Word Clustering. In *Proceedings of IWPT 2009*.

CANDITO M.-H., CRABBÉ B., DENIS P. & GUÉRIN F. (2009). Analyse syntaxique du français : des constituants aux dépendances. In *Actes de TALN*.

CANDITO M.-H., NIVRE J., DENIS P. & HENESTROZA ANGUIANO E. (2010b). Benchmarking of Statistical Dependency Parsers for French. In *Proceedings of COLING'2010*.

CHARNIAK E. & JOHNSON M. (2005). Coarse-to-Fine n-Best Parsing and MaxEnt Discriminative Reranking. In *Proceedings of ACL*.

COLLINS M. (1997). Three Generative, Lexicalised Models for Statistical Parsing. In *Proceedings of the 35th Annual Meeting of the ACL*.

COLLINS M. (2000). Discriminative Reranking for Natural Language Parsing. In *Proceedings of ICML*.

CRABBÉ B. & CANDITO M. (2008). Expériences d'analyse syntaxique du français. In *Actes de TALN*.

CRAMMER K., DEKEL O., KESHET J., SHALEVSHWARTZ S. & SINGER Y. (2006). Online Passive-Aggressive Algorithm. *Journal of Machine Learning Research*.

DENIS P. & SAGOT B. (2010). Exploitation d'une ressource lexicale pour la construction d'un étiqueteur morphosyntaxique état-de-l'art du français. In *Actes de TALN*.

JOHNSON M. & URAL A. E. (2010). Reranking the Berkeley and Brown Parsers. In *Human Language Technologies : The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, p. 665–668, Los Angeles, California : Association for Computational Linguistics.

MATSUZAKI T., MIYAO Y. & ICHI TSUJII J. (2005). Probabilistic CFG with Latent Annotations. In *Proceedings of ACL*.

MCCLOSKY D., CHARNIAK E. & JOHNSON M. (2008). When is Self-Training Effective for Parsing ? In D. SCOTT & H. USZKOREIT, Eds., *COLING*, p. 561–568.

MCDONALD R. (2006). *Discriminative Training and Spanning Tree Algorithms for Dependency Parsing*. PhD thesis, University of Pennsylvania.

MCDONALD R., CRAMMER K. & PEREIRA F. (2005). Online Large-Margin Training of Dependency Parsers. In *Association for Computational Linguistics (ACL)*.

NIVRE J., HALL J., NILSSON J., CHANEV A., ERYIGIT G., KÜBLER S., MARINOV S. & MARSI E. (2007). Maltparser : A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, **13**(2), 95–135.

PETROV S., BARRETT L., THIBAUX R. & KLEIN D. (2006). Learning Accurate, Compact, and Interpretable Tree Annotation. In *ACL*.

PETROV S. & KLEIN D. (2007). Improved Inference for Unlexicalized Parsing. In *HLT-NAACL*, p. 404–411.

PULLUM G. K. & SCHOLZ B. C. (2001). On the distinction between model-theoretic and generative-enumerative syntactic frameworks. In *Logical Aspects of Computational Linguistics*.

RAMBOW O. (2010). The Simple Truth about Dependency and Phrase Structure Representations : An Opinion Piece. In *NAACL HLT*.

ROSENBLATT F. (1958). The Perceptron : A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*.

SEDDAH D., CANDITO M. & CRABBÉ B. (2009). Adaptation de parsers statistiques lexicalisés pour le français : Une évaluation complète sur corpus arborés. In *TALN*.