Prédiction structurée pour l'analyse syntaxique en constituants par transitions : modèles denses et modèles creux

Maximin Coavoux* — **Benoît Crabbé*,****

* *Univ. Paris Diderot, Sorbonne Paris Cité*
Alpage (Inria)
Bât. Olympe de Gouges, 8 Place Paul Ricoeur, 75013 Paris
** *Institut universitaire de France*
maximin.coavoux@inria.fr, bcrabbe@linguist.univ-paris-diderot.fr

RÉSUMÉ. *L'article présente une méthode d'analyse syntaxique en constituants par transitions qui se fonde sur une méthode de pondération des analyses par apprentissage profond. Celle-ci est comparée à une méthode de pondération par perceptron structuré, vue comme plus classique. Nous introduisons tout d'abord un analyseur syntaxique pondéré par un réseau de neurones local et glouton qui s'appuie sur des plongements. Ensuite nous présentons son extension vers un modèle global et à recherche par faisceau. La comparaison avec un modèle d'analyse de la famille perceptron global et en faisceau permet de mettre en évidence les propriétés étonnamment bonnes du modèle neuronal à recherche gloutonne.*

ABSTRACT. *The article introduces a transition-based constituent parsing method relying on a deep-learning weighting scheme. This weighting method is compared to a structured perceptron, which is a more traditional method. First of all, we introduce a syntactic parser weighted by a local greedy neural network based on symbol embeddings. Then, we extend this model to a global beam-search based model. Our experiments highlight the surprisingly good properties of the greedy local neural parser.*

MOTS-CLÉS : *analyse syntaxique en constituants, réseau de neurones artificiels, plongements lexicaux, apprentissage structuré.*

KEYWORDS: *constituent parsing, neural networks, word embeddings, structured learning.*

1. Introduction

Cet article présente une méthode d'analyse syntaxique en constituants par transitions dont la composante de pondération se fonde sur un codage par vecteurs denses des symboles discrets. En traitement automatique des langues, l'utilisation de plongements lexicaux pour représenter les symboles a été popularisée par Collobert et Weston (2008) et Mikolov *et al.* (2013). Si la représentation des mots par des plongements offre la perspective d'une meilleure généralisation pour les modèles statistiques, diverses expériences montrent que l'intégration naïve de plongements acquis sur de grandes masses de données dans des systèmes d'analyse syntaxique existants n'apporte que peu ou pas d'améliorations immédiates (Andreas et Klein, 2014) .

On présente une autre perspective sur le problème observé par Andreas et Klein (2014) en mettant en évidence qu'il est fondamental pour un analyseur lexicalisé d'exprimer les interactions, au sens statistique, entre les plongements. On met en évidence l'importance des interactions en introduisant plusieurs modèles d'analyse syntaxique de complexité croissante pour le cas de la prédiction structurée. Ceux-ci sont formulés dans le paradigme « apprentissage profond » et sont comparés aux modèles d'apprentissage à représentations creuses utilisés traditionnellement depuis Collins (2002) et Lafferty *et al.* (2001).

Ainsi, après un bref rappel du contexte scientifique (section 2), on présente le système d'analyse par transitions et de pondération en section 3. La section 4 introduit une première famille de modèles d'apprentissage locaux pour la prédiction structurée et la section 5 généralise aux modèles d'apprentissage globaux. La section 6 propose des résultats expérimentaux sur le français et l'anglais qui mettent en évidence quelques propriétés générales des modèles à plongements et notamment l'importance de la représentation des interactions entre plongements.

2. Contexte scientifique

2.1. *Modèles lexicalisés et non lexicalisés*

La désambiguïsation est le problème essentiel de l'analyse syntaxique automatique des langues naturelles. On illustre la problématique par un exemple d'attachement de syntagme prépositionnel. En disposant de l'information qu'une *tomate* est un dépendant vraisemblable de *salade*, on préfère l'analyse *Pierre mange (la salade avec des tomates)* alors que l'analyse *Pierre mange (la salade) (avec des couverts)* sera préférée si on a établi que *couverts* est un dépendant plus vraisemblable de *manger* que de *salade*. Cependant, acquérir une telle connaissance bilexicale est en pratique très difficile à partir des corpus arborés de taille relativement modeste sur lesquels sont entraînés les modèles de désambiguïsation.

À partir de là, on peut structurer les modèles d'analyse syntaxique en constituants en deux paradigmes : le paradigme lexicalisé et le paradigme non lexicalisé. Le paradigme non lexicalisé fait le pari qu'il vaut mieux contourner le problème d'acquisition

de connaissances bilexicales, considéré comme trop difficile. L'idée est dans ce cas de construire des modèles qui tirent parti de régularités structurales que l'on peut observer facilement sur les arbres. Par exemple, les groupes nominaux sujets sont plus souvent pronominalisés que les groupes nominaux objets. Ce type de motif se code facilement dans des arbres en constituants en affinant l'annotation existante par de nouvelles annotations latentes. Ce paradigme non lexicalisé a d'abord été développé par Klein et Manning (2003) et a évolué vers PCFG-LA (Petrov *et al.*, 2006) et ses dérivés (Shindo *et al.*, 2012).

Le paradigme lexicalisé, quant à lui, maintient le pari que l'information clé à capturer pour désambiguïser est essentiellement lexicalisée, même si l'information bilexicale est difficile à modéliser. Pour cette raison les modèles lexicalisés génératifs de Collins (1997) et Charniak (2000) élaborent considérablement des solutions qui permettent de décomposer la pondération des règles de grammaire lexicalisées. Ceux-ci développent également des modèles de repli vers des représentations agrégées des mots comme des catégories morphosyntaxiques et/ou des agrégats de Brown (Koo *et al.*, 2008).

Les reformulations des modèles lexicalisés génératifs par des modèles discriminants souvent dérivés de Collins (2002) ont apporté de nouvelles méthodes pour pondérer les règles de grammaire, à l'aide de fonctions indicatrices. Si les fonctions indicatrices permettent de décomposer la pondération des règles en indiquant la présence de mots, de catégories, ou encore de couples ou de triplets de mots, celles-ci ne permettent toutefois pas de compenser les difficultés posées aux méthodes d'estimation statistique par la taille et la distribution zipfienne du vocabulaire. Ainsi, ce système de représentations se heurte toujours à des problèmes de dispersion des données à partir des corpus arborés utilisés.

2.2. *Plongements lexicaux*

La représentation des mots par des plongements, c'est-à-dire des vecteurs de réels de faible dimensionnalité, offre la perspective d'une meilleure généralisation pour les modèles lexicalisés, comme cela a été montré pour d'autres tâches que l'analyse syntaxique (Bengio *et al.*, 2003 ; Collobert et Weston, 2008). En effet, dans un cas idéal, de tels plongements structurent implicitement le lexique en donnant à des symboles similaires (par exemple des symboles de mots ou de catégories syntaxiques) des représentations similaires. Cela permet de limiter les effets de la dispersion des données en généralisant les informations apprises sur un symbole aux symboles qui lui sont similaires (Bengio *et al.*, 2003).

Cependant, diverses expériences montrent que l'intégration naïve de plongements acquis sur de grandes masses de données dans des systèmes d'analyse syntaxique existants n'apporte que peu ou pas d'améliorations immédiates (Andreas et Klein 2014). D'après ces auteurs, une explication semble venir du fait que les informations données par les plongements lexicaux sont redondantes avec d'autres informations connues par

ailleurs comme les catégories morphosyntaxiques. Cet article apporte un autre élément de réponse à cette question en mettant en évidence qu'il est fondamental pour un analyseur lexicalisé d'exprimer les interactions, au sens statistique, entre les plongements. Ces interactions sont cruciales vu qu'elles modélisent des notions telles que des relations de dépendances, ou des règles de grammaire. On met en évidence l'importance des interactions en introduisant plusieurs modèles d'analyse syntaxique de complexité croissante pour le cas de la prédiction structurée (section 4.2).

2.3. *Analyse syntaxique et réseaux de neurones*

En analyse en constituants, Henderson (2003) est un des premiers à utiliser des réseaux de neurones récurrents pour donner une représentation finie à un historique arbitrairement long de décisions de *parsing*. Notons que cet analyseur prédate l'arrivée du *deep learning* et ne s'appuie pas sur des plongements lexicaux.

Plus récemment, Chen et Manning (2014) ont proposé un réseau de neurones simple pour l'analyse syntaxique en dépendances par transitions, qui s'appuie sur des plongements de mots, de catégories morphosyntaxiques et de relations de dépendances. Deux propositions dérivées de cet article nous intéressent particulièrement. Celles-ci tentent de généraliser le réseau de Chen et Manning (2014) au cas de l'apprentissage structuré. Dyer *et al.* (2015) – analyse en dépendances – et Watanabe et Sumita (2015) – analyse en constituants – utilisent des réseaux de neurones récurrents pour améliorer les représentations de l'historique des décisions et de la file d'attente. Weiss *et al.* (2015) utilisent les couches cachées du réseau de neurones pour servir d'entrée à un perceptron structuré standard (Collins, 2002).

Le modèle de pondération que nous utilisons est adapté de Chen et Manning (2014) pour l'analyse en constituants et reprend la formulation de Coavoux et Crabbé (2015). En particulier, nous proposons deux extensions par rapport à Coavoux et Crabbé (2015). Premièrement, nous utilisons des informations morphologiques pour pondérer les transitions, en représentant chaque caractéristique par un vecteur de réels, comme cela est fait généralement pour les mots. Notons que si de telles informations sont fréquemment utilisées par des analyseurs en constituants, du fait notamment de la disponibilité du jeu de données SPMRL (Seddah *et al.*, 2013), aucun analyseur fondé sur un réseau de neurones n'utilise de tels attributs comme source d'information. Deuxièmement, nous adaptons le modèle de Weiss *et al.* (2015), proposé initialement pour l'analyse en dépendances, ce qui permet de donner un nouveau repère pour l'analyse en constituants.

3. Analyse syntaxique en constituants par décalage réduction

Dans ce qui suit, nous nous intéressons au paradigme lexicalisé, et en particulier à l'analyse syntaxique par transitions. En ce domaine, les principales propositions dérivent de Sagae et Lavie (2006). Elles s'appuient généralement sur un perceptron

structuré (Collins, 2002) et utilisent la recherche par faisceau pour le décodage (Zhang et Clark, 2009 ; Zhu *et al.*, 2013 ; Crabbé, 2014), ce qui permet un temps d'exécution linéaire en la taille de la phrase à analyser.

3.1. *Représentation et induction d'une grammaire robuste*

L'analyseur que nous présentons suppose une grammaire de constituants binarisée dont les règles sont lexicalisées (2-LCFG). Les règles d'une grammaire 2-LCFG ont une des formes suivantes :

$$\begin{array}{l} A[h] \rightarrow B[b]\, C[c] \quad \text{où } h \in \{b, c\} \\ A[h] \rightarrow h \end{array} \qquad [1]$$

où $A, B, C \in D$ sont des symboles délexicalisés, $h, b, c \in T$ sont des symboles terminaux et où l'ensemble des symboles non terminaux de la grammaire est $N \subseteq D \times T$.

Dans ce qui suit, on suppose un corpus arboré dont les arbres syntagmatiques sont lexicalisés. La lexicalisation est obtenue dans notre cas par la procédure décrite par Crabbé (2015). La grammaire est extraite depuis ce corpus arboré enrichi en utilisant une transformation des arbres en forme normale de Chomsky approximative que Klein et Manning (2003) appellent markovization horizontale d'ordre 0. La procédure de binarisation est suivie d'une procédure de réduction des productions unaires. Cette dernière ne s'applique toutefois pas aux productions unaires qui introduisent des nœuds terminaux ou préterminaux. Ce dernier choix permet de faciliter l'interface entre l'analyseur et un étiqueteur morphosyntaxique sans exiger de transformer le jeu de catégories morphosyntaxiques.

On remarque toutefois que notre procédure de transformation des données garantit non seulement des règles de la forme [1] mais introduit également des règles de la forme :

$$\begin{array}{l} A[h] \rightarrow B[b]\, c \\ A[h] \rightarrow c\, B[b] \quad \text{où } h \in \{b, c\}. \end{array} \qquad [2]$$

Enfin, on extrait uniquement D du corpus arboré ainsi transformé. La grammaire complètement lexicalisée est laissée implicite. C'est le système de transitions qui est responsable de l'instanciation dynamique des symboles lexicalisés.

3.2. *Système de transitions*

L'analyseur syntaxique est un analyseur à décalage réduction (*shift-reduce*) organisé autour de deux structures de données. La première est une pile de symboles $\mathbf{S} = \ldots|s_2|s_1|s_0$ de sommet s_0. Les symboles empilés sont soit des terminaux soit des non-terminaux lexicalisés de la forme $A[x]$. La seconde est une file de terminaux $T = t_1 \ldots t_n$. L'analyse syntaxique est réalisée en générant successivement des configurations C de la forme $\langle j, \mathbf{S}, \cdot \rangle$ où $\mathbf{S}$ dénote la pile et j l'index du premier élément de la file. Étant donnée une configuration de départ $C_0 = \langle 1, \epsilon, \perp \rangle$, une

étape de dérivation $C_{t-1} \overset{a_{t-1}}{\Rightarrow} C_t$ génère une nouvelle configuration C_t en appliquant une action $a_{t-1} \in A$ prise dans l'ensemble des actions possibles. Il y a une action $RL(X), RR(X), RU(X)$ correspondant à chaque non-terminal délexicalisé $X \in D$. Autrement dit, $|A| = 3D + 2$. On décrit les schémas d'actions en figure 1 en notation déductive étendue (Nederhof, 2003). Remarquons qu'une dérivation est terminée avec succès lorsque la configuration C_{3n-1} est engendrée. Une séquence de dérivation $C_{0 \Rightarrow \tau}$ est la séquence $C_0 \overset{a_0}{\Rightarrow} \ldots \overset{a_{\tau-1}}{\Rightarrow} C_\tau$.

Finalement, on peut démontrer par induction que pour un analyseur par décalage réduction et une grammaire en forme normale de Chomsky, le nombre de transitions τ de toute analyse d'une phrase de longueur n est $2n - 1$ et que l'introduction des règles de la forme [2] autorise que $2n - 1 \leq \tau \leq 3n - 1$. Comme illustré par Crabbé (2014), le système de pondération que nous utilisons présente une forte corrélation entre longueur des hypothèses et poids des analyses. Par conséquent nous introduisons une action de réduction fantôme (GR) qui contraint que $\tau = 3n - 1$.

INIT	$\langle 1, \epsilon, \bot \rangle : 0$
GOAL	$\langle n+1, \tau, \bot \rangle : w$

SHIFT	$\frac{\langle j, \mathbf{S}, \bot \rangle : w}{\langle j+1, \mathbf{S} \mid t_j.tag[t_j.word]), \top \rangle : w + f(S, \langle j, \mathbf{S} \rangle)}$
RL(X)	$\frac{\langle j, \mathbf{S}_\ominus \mid c_1[t_1]\ c_0[t_0], \bot \rangle : w}{\langle j, \mathbf{S}_\ominus \mid X[t_1], \bot \rangle : w + f(RL(X), \langle j, \mathbf{S} \rangle)}$
RR(X)	$\frac{\langle j, \mathbf{S}_\ominus \mid c_1[t_1]\ c_0[t_0], \bot \rangle : w}{\langle j, \mathbf{S}_\ominus \mid X[t_0], \bot \rangle : w + f(RR(X), \langle j, \mathbf{S} \rangle)}$
RU(X)	$\frac{\langle j, \mathbf{S}_\ominus \mid c_0[t_0], \top \rangle : w}{\langle j, \mathbf{S}_\ominus \mid X[t_0], \bot \rangle : w + f(RU(X), \langle j, \mathbf{S} \rangle)}$
GR(X)	$\frac{\langle j, \mathbf{S}_\ominus \mid c_0[t_0], \top \rangle : w}{\langle j, \mathbf{S}_\ominus \mid c_0[t_0], \bot \rangle : w + f(GR, \langle j, \mathbf{S} \rangle)}$

Figure 1. *Schémas de règles d'inférence pour l'analyse par transitions*

3.3. *Gestion du non-déterminisme et pondération des analyses*

L'algorithme n'est pas naturellement déterministe. Pour choisir une dérivation, il s'appuie sur une fonction de pondération qui donne un poids à chaque (sous-) dérivation $C_{0 \Rightarrow \tau}$ possible. Par convention, la dérivation à choisir en fin d'analyse est celle de poids maximal. Les différentes méthodes de pondération utilisées dans cet article s'appuient systématiquement sur une fonction de scorage additive de la forme :

$$W(C_{0 \Rightarrow \tau}) = \sum_{t=0}^{\tau-1} f(a_t, C_t). \qquad [3]$$

Autrement dit f est une fonction de scorage qui dépend de l'action a à effectuer et d'informations accessibles localement à la configuration C. On suppose donc que le score d'une dérivation se décompose comme la somme des scores de chacun de ses pas. Cela permet de coupler le mécanisme de pondération avec les règles d'inférence (figure 1). Dans ce contexte, le problème de l'analyse syntaxique consiste à trouver la meilleure dérivation possible de la phrase c'est-à-dire à donner la solution de :

$$\hat{C}_{0\Rightarrow 3n-1} = \underset{C_{0\Rightarrow 3n-1}\in \text{GEN}_{3n-1}}{\text{argmax}} \; W(C_{0\Rightarrow 3n-1}) \qquad [4]$$

où GEN_{3n-1} est l'ensemble des analyses possibles pour une phrase de longueur n. En pratique on utilise une recherche approximative avec un faisceau de taille K à chaque étape temporelle : seules les K meilleures configurations à une étape t sont utilisées pour poursuivre la recherche de solutions à l'étape $t+1$.

L'essentiel de cet article porte sur la mise au point d'une fonction paramétrique de scorage f et sur l'estimation des paramètres de cette fonction à partir d'un corpus arboré binarisé. On décrit en section 4 des modèles d'apprentissage locaux et en section 5 un modèle d'apprentissage global.

4. Modèles d'apprentissage local pour l'analyse syntaxique

Dans le cas de l'analyse syntaxique, les objets à prédire sont des objets structurés. Pour une phrase donnée, l'analyseur doit déterminer la meilleure séquence d'actions. Dans un premier temps, une simplification consiste à ignorer la structure des phrases et à entraîner un classifieur à prédire des actions individuelles à partir des configurations de référence. Le corpus d'entraînement est alors considéré comme un ensemble $\mathcal{D} = \{(C_i, a_i)\}_{i\in\{1,\ldots N\}}$ de couples composés d'une configuration et d'une action de référence. On appelle modèle *local*, un modèle d'analyse qui fait cette hypothèse simplificatrice.

Cette section s'intéresse à la fonction de scorage $f(\cdot,\cdot)$ présentée en équation 3 et aux représentations sous-jacentes. Nous commençons par motiver le passage d'un modèle de représentations creuses fondé sur des fonctions indicatrices à un modèle de représentations denses. Ensuite, nous proposons d'adapter le modèle statistique sous-jacent à des représentations denses. Pour cela, nous instancions plusieurs modèles locaux sous la forme d'architectures de réseaux de neurones artificiels de complexité croissante destinées à isoler les différentes contributions des représentations denses et du nouveau modèle statistique.

4.1. *Des représentations creuses aux représentations denses*

Dans le modèle général de pondération présenté en équation 3, la fonction f s'appuie implicitement sur une représentation vectorielle d'une configuration, que nous notons $\mathbf{\Phi}(C)$. Dans les modèles discriminants linéaires traditionnellement utilisés,

$f(a, C)$ se formule à partir du produit scalaire entre un vecteur de poids $\mathbf{w}_a$ et un vecteur de traits booléens $\mathbf{\Phi}(C)$. Dans le cas du perceptron, $f(a, C) = \mathbf{w}_a \cdot \mathbf{\Phi}(C)$ est un simple produit scalaire. Dans le cas de la régression logistique multinomiale, qui nous intéresse plus particulièrement :

$$f(a, C) = \log \left(\frac{\exp(\mathbf{w}_a \cdot \mathbf{\Phi}(C))}{\sum_{a' \in A} \exp(\mathbf{w}_{a'} \cdot \mathbf{\Phi}(C))} \right) = \log P(a \mid C; \mathbf{W}) \qquad [5]$$

où $\mathbf{W}$ est la matrice des paramètres dont chaque ligne est le vecteur de poids $\mathbf{w}_a$ correspondant à une action a. On note que l'utilisation de log-probabilités permet de conserver le modèle additif présenté en équation 3. Ce paragraphe s'intéresse à la fonction de représentation $\mathbf{\Phi}$. On commence par présenter les fonctions indicatrices habituellement utilisées pour valuer $\mathbf{\Phi}$ ainsi que les limites de ce paradigme. On introduit ensuite des représentations denses en donnant une nouvelle formulation à $\mathbf{\Phi}$ et on montre que cela ne change pas fondamentalement l'algorithme d'analyse.

4.1.1. *Représentations creuses et fonctions indicatrices*

La fonction $\mathbf{\Phi}$ est définie traditionnellement à l'aide de fonctions indicatrices valuées à partir de l'information contenue dans une configuration C. Généralement, on se limite à de l'information locale, c'est-à-dire que les fonctions indicatrices ne peuvent porter que sur les premiers éléments de la pile et de la file d'attente. On donne en figure 2 un schéma des informations accessibles depuis une configuration. Les notations utilisées permettent d'adresser les différentes positions de cette configuration. Les symboles s_i indexent la position i d'un élément de la pile ; les symboles c et w dénotent respectivement un non-terminal et le terminal qui le lexicalise. Enfin, les lettres t(op), l(eft) et r(right) en indices signifient qu'il s'agit de la racine d'un élément de la pile, de son fils gauche ou de son fils droit. Par exemple, $s_0.c_l$ dénote le non-terminal qui est le fils gauche du premier élément de la pile. Sur la file d'attente, les terminaux sont adressés à partir de leur position dans la phrase j, $j+1$, $j+2$, $j+3$ correspondant à q_0, q_1, q_2, q_3. Notons que les terminaux sont eux-mêmes structurés. *A minima*, chaque terminal est constitué d'un mot-forme et d'une étiquette morphosyntaxique. On utilisera $q_i.w$ pour désigner le mot-forme et $q_i.tag$ pour l'étiquette morphosyntaxique.

Pour valuer $\mathbf{\Phi}$, les fonctions indicatrices peuvent prendre, en principe, la forme de n'importe quel prédicat portant sur un symbole de la figure 2. Toutefois, en pratique, elles consistent en la conjonction de 1 à 3 conditions, par exemple :

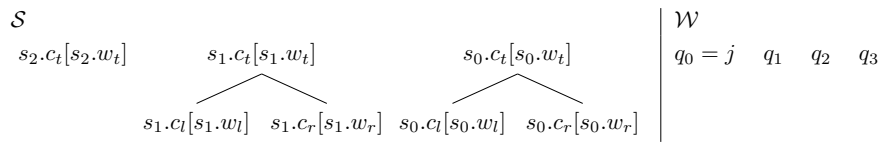

Figure 2. *Information accessible aux fonctions de traits depuis une configuration C*

$$\phi_k(C) = \begin{cases} 1 & \textbf{si} \text{ le mot tête en sommet de pile } (s_0.w_t.w) \text{ est } \textit{chat} \\ 0 & \textbf{sinon} \end{cases}$$

$$\phi_{k'}(C) = \begin{cases} 1 & \textbf{si} \text{ le mot tête en sommet de pile } (s_0.w_t.w) \text{ est } \textit{chat} \textbf{ et} \text{ que le symbole} \\ & \text{grammatical } (s_1.c_t) \text{ en sous-sommet de pile est } \textit{Déterminant} \\ 0 & \textbf{sinon} \end{cases}$$

L'utilisation de conjonctions de conditions permet de modéliser des interactions entre les différents symboles, et de capturer par exemple des préférences sélectionnelles. Dans ce cadre, $\mathbf{\Phi}(C) \in \{0,1\}^d$ est un vecteur très creux de très grande dimensionnalité. Seuls quelques-uns de ses coefficients ne sont pas nuls.

Ce type de représentations est utilisé avec succès dans la plupart des tâches de TAL reposant sur l'apprentissage statistique, mais est également sujet à plusieurs limitations. D'une part, les corpus arborés sont petits et les mots suivent une distribution zipfienne. La plupart des mots n'apparaissent qu'une ou deux fois dans un corpus, de sorte qu'il est très difficile d'estimer de manière fiable les statistiques qui leur sont liées, et plus encore celles liées aux couples voire aux triplets de mots (par exemple pour modéliser des préférences de sélection). D'autre part, les fonctions indicatrices codent chaque information lexicale sur une dimension distincte de $\mathbf{\Phi}$. Cela revient à donner au lexique une représentation non structurée. Au contraire, le passage à des représentations continues de type plongement, permet de tirer parti de similarités entre symboles (cf. section 2.2) pour atténuer la dispersion des données (Bengio *et al.*, 2003).[1]

4.1.2. *Représentations denses*

Partant de ces observations, nous voudrions construire un modèle entièrement fondé sur des représentations denses de symboles. Concrètement, ce changement peut s'exprimer en redéfinissant $\mathbf{\Phi}(C)$ non plus à l'aide de fonctions caractéristiques, mais à partir d'un dictionnaire $\delta : \Sigma \rightarrow \mathbb{R}^d$ qui associe un plongement à d dimensions (cette fois, d est supposé petit) à chaque symbole discret. En notant x_1, x_2, …, x_n les symboles discrets correspondant chacun à une position de la figure 2, on pose :

$$\mathbf{\Phi}(C) = [\delta(x_1); \delta(x_2); \ldots; \delta(x_n)].$$

On note que ce changement est transparent pour la méthode de pondération présentée en équation 5 : pour l'instant, la seule modification apportée par l'introduction de symboles denses relève de la représentation interne de $\mathbf{\Phi}$. Si l'on suppose que δ renvoie le codage *one-hot* [2] d'un symbole discret, on retrouve une représentation équivalente à celle obtenue par fonctions indicatrices, dans le cas où l'on n'utilise que des fonctions indicatrices à une seule condition. On identifie ainsi deux propriétés clés des représentations denses. D'une part, on peut donner à un symbole une représentation plus

1. Dans le modèle que l'on présente, cette similarité sera entièrement inférée à partir des données d'entraînement.

2. Le codage *one-hot* du i-ième élément d'un ensemble est le vecteur booléen dont le i-ième coefficient vaut 1 et tous les autres 0.

expressive que son codage *one-hot*. Néanmoins, on ne dispose pas, pour l'instant, de méthode pour tenir compte des interactions entre symboles qui sont modélisées par des couples ou des triplets de conditions dans le cas creux.

On a montré dans cette section que le passage d'un modèle creux à un modèle dense ne modifie pas fondamentalement l'algorithme d'analyse. Il nous reste à répondre aux questions suivantes. D'où vient la fonction δ et quelles propriétés est-elle supposée satisfaire ? Comment exprimer, pour le cas dense, une contrepartie des interactions entre variables exprimées naturellement dans le cas creux par des conjonctions de conditions ? Pour répondre à ces questions, on présente dans la section suivante plusieurs modèles de pondération sous la forme de réseaux de neurones à propagation avant.

4.2. *Architectures de réseaux de neurones artificiels*

Nous présentons dans cette section plusieurs architectures de réseaux de neurones. Ces architectures sont de complexité croissante et permettent d'isoler les contributions de deux modifications de la fonction de scorage $f(\cdot, \cdot)$. La première vise à introduire des paramètres partagés sous la forme de plongements. La seconde cherche à modéliser les interactions entre variables à l'aide de couches cachées non linéaires. Notons que ces modèles ont déjà été présentés dans (Coavoux et Crabbé, 2015). Nous les exposons ici pour en présenter deux extensions : l'une consiste à utiliser des informations morphologiques additionnelles pour la pondération des analyses (section 4.2.4) ; l'autre est une généralisation au cas de l'apprentissage structuré (section 5).

Commençons par expliciter la forme que prend le dictionnaire δ postulé dans la section précédente. On suppose que la pondération des analyses peut s'appuyer sur l'ensemble des symboles discrets représentés sur la figure 2. Pour une configuration donnée, ces symboles sont instanciés par des mots-formes, des catégories syntagmatiques et des parties du discours pour former un vecteur $x_1, x_2, \ldots, x_n$ de symboles discrets. Chaque symbole de ce vecteur est typé par un type t indiquant s'il s'agit d'un mot-forme (W), d'une catégorie syntagmatique (S) ou d'une partie du discours (T). Ce typage nous permet de distinguer 3 dictionnaires $\delta_t : \Sigma_t \rightarrow \mathbb{R}^{d_t}$ de manière à pouvoir utiliser des plongements de plus petite dimensionnalité pour les catégories syntaxiques (de l'ordre de la centaine de symboles) que pour les mots-formes (de l'ordre de la dizaine de milliers de symboles pour le *French Treebank* (Abeillé *et al.*, 2003)).

Pour chaque type, le dictionnaire δ_t est représenté par une matrice $\mathbf{E}^{(t)} \in \mathbb{R}^{d_t \times |t|}$ dont chaque colonne contient le plongement d'un symbole. En notant $\mathbf{e}_x \in \mathbb{R}^{|t|}$ la représentation *one-hot* du symbole x de type t, on obtient son plongement $\mathbf{e}_x^{(t)}$ par multiplication matricielle : $\mathbf{e}_x^{(t)} = \mathbf{E}^{(t)} \cdot \mathbf{e}_x$. En faisant apparaître les dictionnaires sous la forme des matrices $\mathbf{E}^{(t)}$, on met en évidence qu'ils ne sont pas donnés *a priori* mais qu'il s'agit de paramètres que le modèle doit estimer au même titre que la matrice de poids $\mathbf{W}$ de l'équation 5.

Les modèles que nous présentons modélisent tous les probabilités $P(a|C;\boldsymbol{\theta})$ de manière à pondérer chaque action a par $\log P(a|C;\boldsymbol{\theta})$. Ils diffèrent cependant par leur façon de calculer cette quantité et par le nombre de paramètres $\boldsymbol{\theta}$ manipulés.

4.2.1. *Un modèle de base : la régression logistique multinomiale*

Le premier modèle que nous nommons RLM nous sert de *baseline*. Il s'agit d'une régression logistique multinomiale que l'on peut voir comme un cas particulier d'un réseau de neurones à une couche (figure 3, partie gauche). Dans ce paragraphe et dans les suivants, nous noterons $\mathbf{h}^{(i-1)}$ la i-ième couche d'un réseau de neurones et $\mathbf{h}^{(s)}$ sa couche de sortie. Dans le modèle RLM, la probabilité conditionnelle d'une action se calcule par les opérations suivantes :

$$\mathbf{h}^{(s)} = \text{softmax}\left(\mathbf{W} \cdot \begin{bmatrix} \mathbf{e}_{x_1} \\ \mathbf{e}_{x_2} \\ \vdots \\ \mathbf{e}_{x_n} \end{bmatrix} + \mathbf{b}\right)$$

$$P(a|C,\boldsymbol{\theta}) = \mathbf{h}^{(s)}_a.$$

Ce modèle s'appuie sur des représentations creuses de symboles mais ne prend pas d'interactions en compte. Il a pour paramètres $\boldsymbol{\theta} = (\mathbf{W}, \mathbf{b})$.

4.2.2. *Introduction des plongements partagés : réseau de neurones linéaire*

Dans le second modèle, nommé DENSE LIN par la suite, nous introduisons explicitement des paramètres partagés sous la forme de plongements. Pour cela, on ajoute une couche intermédiaire (figure 3, partie centrale) qui consiste en la concaténation des représentations denses de symboles. La probabilité conditionnelle d'une action prend la forme suivante :

$$\mathbf{h}^{(0)} = \begin{bmatrix} \mathbf{e}^{(t_1)}_{x_1} \\ \mathbf{e}^{(t_2)}_{x_2} \\ \vdots \\ \mathbf{e}^{(t_n)}_{x_n} \end{bmatrix} \qquad \text{(opération de } \textit{lookup}\text{)}$$

$$\mathbf{h}^{(s)} = \text{softmax}\left(\mathbf{W} \cdot \mathbf{h}^{(0)} + \mathbf{b}\right)$$

$$P(a|C,\boldsymbol{\theta}) = \mathbf{h}^{(s)}_a.$$

Comme le modèle RLM, il s'agit d'un modèle linéaire qui ne prend pas en compte d'interactions entre variables. Il ne permet pas d'apprendre de fonctions de décision plus complexes que le modèle RLM. On peut voir ce modèle comme une extension de la régression logistique multinomiale avec des paramètres partagés. Un même symbole peut apparaître dans plusieurs positions du vecteur d'entrée $x_1, x_2, \ldots x_n$. Avec les plongements, une partie des paramètres liés à un symbole ne dépend plus de sa

position. Les paramètres à estimer contiennent également les trois matrices de plongements : $\boldsymbol{\theta} = (\mathbf{E}^{(\mathrm{T})}, \mathbf{E}^{(\mathrm{W})}, \mathbf{E}^{(\mathrm{S})}, \mathbf{W}, \mathbf{b})$.

4.2.3. *Introduction d'interactions entre variables : réseaux de neurones profonds non linéaires*

Le troisième modèle cherche à modéliser des interactions entre les symboles. Nous avons vu en section 4.1.1 que les fonctions indicatrices définies à l'aide de plusieurs conditions sur une configuration permettaient de modéliser des interactions entre les symboles (par exemple des préférences sélectionnelles). Toutefois, le nombre de fonctions indicatrices à définir pour capturer toutes les interactions est très grand et les considérer toutes est généralement redondant. Dans le cas dense, la formulation d'interactions n'est pas aussi immédiate. Or, nous faisons l'hypothèse que la prise en compte de ces interactions est cruciale pour tirer parti des informations données par les plongements.

Dans le cadre des réseaux de neurones, on propose de formuler les interactions entre variables à l'aide de couches cachées additionnelles non linéaires. Le calcul de la probabilité conditionnelle d'une action prend la forme suivante :

$$\mathbf{h}^{(0)} = \begin{bmatrix} \mathbf{e}_{x_1}^{(t_1)} \\ \mathbf{e}_{x_2}^{(t_2)} \\ \vdots \\ \mathbf{e}_{x_n}^{(t_n)} \end{bmatrix} \qquad \text{(opération de } \textit{lookup}\text{)}$$

$$\mathbf{h}^{(i+1)} = g\left(\mathbf{W}^{(i)} \cdot \mathbf{h}^{(i-1)} + \mathbf{b}^{(i)}\right) \qquad \text{(pour chaque couche cachée)}$$

$$\mathbf{h}^{(s)} = \text{softmax}\left(\mathbf{W}^{(s)} \cdot \mathbf{h}^{(s-1)} + \mathbf{b}^{(s)}\right)$$

$$P(a|C, \boldsymbol{\theta}) = \mathbf{h}_a^{(s)}$$

où $\mathbf{h}^{(s)}$ désigne la couche de sortie et g désigne une fonction non linéaire appliquée terme à terme. Cette fonction est un hyperparamètre du modèle. Chen et Manning (2014) proposent d'utiliser la fonction cube pour exprimer les interactions statistiques entre variables, une unité de la couche cachée est alors une somme pondérée de tous les produits de trois variables possibles. À la place, nous avons utilisé la fonction RELU (*Rectified Linear Unit*) $x \mapsto \max(0, x)$, plus standard, et qui a donné des résultats similaires à la fonction cube lors d'expériences préliminaires.

La première couche du réseau $\mathbf{h}^{(0)}$ consiste, comme dans le modèle précédent, en la concaténation des plongements des symboles $x_1, x_2, \ldots, x_n$. Les couches cachées suivantes peuvent être vues comme des extracteurs de traits latents. Le nombre de couches cachées est un hyperparamètre du modèle. En pratique, nos expériences utilisent seulement une (modèle PROFOND-1), ou deux (modèle PROFOND-2) couches cachées. Le réseau à une couche cachée est un réseau à propagation avant classique. L'utilisation d'un réseau à deux couches per-

met de tester une architecture similaire à celle de Weiss *et al.* (2015) pour le modèle d'apprentissage structuré (section 5)[3]. Le modèle PROFOND-1 a pour paramètres $\boldsymbol{\theta} = (\mathbf{E}^{(\mathrm{T})}, \mathbf{E}^{(\mathrm{W})}, \mathbf{E}^{(\mathrm{S})}, \mathbf{W}^{(1)}, \mathbf{W}^{(s)}, \mathbf{b}^{(1)}, \mathbf{b}^{(s)})$. Le modèle PROFOND-2 doit estimer en plus $\mathbf{W}^{(2)}$ et $\mathbf{b}^{(2)}$.

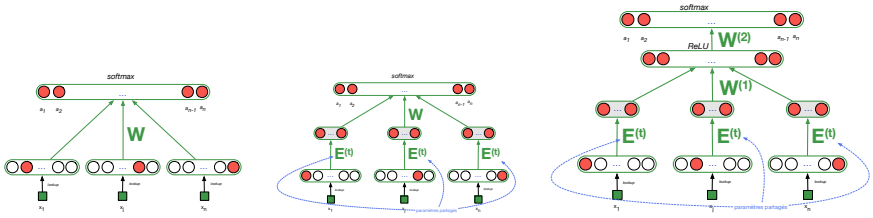

Figure 3. *Architectures de réseaux pour la prédiction locale. Les biais ne sont pas représentés sur les schémas.*

4.2.4. *Intégration d'informations morphologiques dans les modèles profonds*

Pour certaines langues, des informations morphologiques sont utiles pour pondérer les différentes analyses possibles. Les modèles présentés jusqu'ici n'en tiennent pas compte. Nous proposons une méthode directe pour intégrer des informations morphologiques aux modèles denses non linéaires. En section 4.1.1 nous avons vu que les terminaux sont structurés et constitués d'un mot-forme et d'une étiquette morphosyntaxique. Supposons, comme présenté dans Crabbé (2015), que chaque terminal encode une valeur pour chacun des attributs suivants : GENRE (masculin, féminin ou NA), NOMBRE (singulier, pluriel ou NA), MODE (infinitif, subjonctif, impératif, participe, indicatif ou NA) et MWE. L'attribut MWE (pour *multi word expression*) est valué par une étiquette de type BIO (ou *Begin*, *Inside*, *Outside*) indiquant si le mot-forme marque le début d'une expression polylexicale et la catégorie de cette expression (par exemple B_NC+ pour début de nom composé), s'il est à l'intérieur d'une expression polylexicale (I) ou s'il n'est pas concerné par ces cas (O). Notons que dans nos expériences, toutes ces informations sont prédites de manière conjointe par un étiqueteur morphosyntaxique.

Dans le cas creux, on utiliserait de nouvelles fonctions indicatrices pour prendre en compte ces informations (par exemple, des fonctions indicatrices testant l'identité de l'attribut NOMBRE de plusieurs symboles). Dans le cas dense, nous prenons le parti de représenter ces informations à l'aide de plongements de la même manière que les mots-formes ou les catégories syntaxiques. Nous utilisons des plongements à un seul coefficient pour les attributs binaires (GENRE et NOMBRE), des plongements à quatre coefficients pour le mode, et pour l'attribut MWE des plongements de taille identique

3. La justification théorique d'une architecture à deux couches cachées reste à établir. Voir (Montufar *et al.*, 2014) pour un début de justification.

function ASGD($\boldsymbol{\theta}, ((C^{(1)}, a^{(1)}) \dots (C^{(N)}, a^{(N)}), E, \alpha_0, \gamma)$
  $\boldsymbol{\theta} \leftarrow Random$
  $\bar{\boldsymbol{\theta}} \leftarrow \mathbf{0}$
  $t \leftarrow 0$
  for $e = 1$ **to** E **do** ▷ Itérations sur E époques
    for $i = 1$ **to** N **do** ▷ Itérations sur les données
      $\alpha_t \leftarrow \alpha_0/(1 + t \times \gamma)$
      $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha_t \nabla_{\boldsymbol{\theta}} \left(-\log P(a^{(i)}|C^{(i)}; \boldsymbol{\theta}) \right)$
      $\bar{\boldsymbol{\theta}} \leftarrow \bar{\boldsymbol{\theta}} + \boldsymbol{\theta}$
      $t \leftarrow t + 1$
  return $\bar{\boldsymbol{\theta}}/(N \times E)$

Figure 4. *Descente de gradient stochastique moyennée*

à ceux des catégories syntaxiques. Par la suite, nous nommerons PROFOND-1+M et PROFOND-2+M les modèles profonds qui utilisent ces informations morphologiques.

4.3. *Estimation des paramètres*

Les modèles locaux sont entraînés à partir d'un ensemble de couples $\mathcal{D} = \{(C_i, a_i)\}_{i \in \{1,\dots N\}}$ composés d'une configuration et d'une action de référence. Cet ensemble est extrait du corpus arboré de manière déterministe. Pour estimer les paramètres de ces modèles, on doit minimiser la fonction objective suivante :

$$\mathcal{L}(\mathcal{D}, \boldsymbol{\theta}) = -\frac{1}{N} \sum_{i=1}^{N} \log P(a^{(i)}|C^{(i)}; \boldsymbol{\theta}).$$

Cela revient à maximiser la vraisemblance des données observées en supposant ces données indépendantes et identiquement distribuées.

Nous avons utilisé la descente de gradient stochastique moyennée, (ASGD, Polyak et Juditsky, 1992) pour optimiser cet objectif. Nous présentons cet algorithme en figure 4. Le pas d'apprentissage α est ajusté au fur et à mesure des itérations à l'aide de la stratégie de *Power Scheduling*. Sa décroissance dépend d'une constante γ. Nous avons également testé la descente de gradient stochastique standard et l'algorithme ADAGRAD (Duchi *et al.*, 2011). Empiriquement, nous avons observé qu'ASGD permettait d'estimer des paramètres qui généralisaient beaucoup mieux que ces deux autres méthodes. Les différences de résultat avec Coavoux et Crabbé (2015) viennent principalement de cette méthode d'optimisation.

Les modèles neuronaux ont de nombreux hyperparamètres à ajuster (le pas d'apprentissage, la constante γ, le nombre d'itérations, la taille des plongements de chaque type, le nombre de couches cachées et leur taille, la fonction d'activation non linéaire). L'espace de recherche est trop grand pour tester les hyperparamètres de manière exhaustive. Pour mener les expériences que nous allons présenter, nous sommes partis

– pour chacun des modèles – d'une combinaison d'hyperparamètres qui ont donné de bons résultats lors d'expériences préliminaires. Puis nous avons fixé certains hyperparamètres (nombre d'itérations, taille des plongements) et avons fait varier les plus importants (notamment le pas d'apprentissage et la taille de la couche cachée).

5. Prédiction structurée

Comme formalisé par Collins (2002) et Lafferty *et al.* (2001), et comme discuté en section 6 les modèles locaux utilisés dans le cas de la prédiction de structure présentent potentiellement des problèmes de biais locaux. Il est possible de généraliser les modèles locaux au cas structuré pour en faire des modèles destinés à prédire globalement des dérivations. Définir un modèle de prédiction structurée global avec normalisation de type CRF (Lafferty *et al.*, 2001) pose la difficulté qu'il faut pouvoir normaliser les différents scores. Il est tout à fait envisageable d'utiliser des méthodes de normalisation approximatives (Hinton *et al.*, 2006) mais nous présentons ici une reformulation de nos modèles locaux à l'aide du perceptron structuré (Collins, 2002), qui ne nécessite pas de normalisation.

Le modèle global utilise toujours [3] pour réaliser les prédictions. La fonction $f(a, C)$ est instanciée dans le cas présent par un produit scalaire :

$$f(a, C) = \mathbf{w}_a \cdot \mathbf{\Phi}(C) \quad [6]$$

où $\mathbf{\Phi}(C) \in \mathbb{R}^n$ est un vecteur de traits et $\mathbf{w}_a \in \mathbb{R}^n$ un vecteur de paramètres.

Dans le cas structuré, ce qui change c'est la méthode d'estimation des paramètres. On suppose dans ce cas-ci un jeu de données d'entraînement de N phrases, $T = ((\mathcal{T}_1, \mathcal{R}_1), \ldots, (\mathcal{T}_N, \mathcal{R}_N))$ pour lesquelles nous disposons de l'analyse correcte $\mathcal{R}_i$. L'algorithme du perceptron structuré itère sur les données un nombre prédéterminé d'époques. Pour chaque exemple, il compare la meilleure séquence d'actions prédite, étant donnée la valeur courante du vecteur de poids $\mathbf{w}$, avec la séquence correspondant à l'analyse correcte. En cas d'erreur, l'algorithme met à jour les poids en pénalisant ceux associés à l'analyse prédite et en favorisant ceux associés à l'analyse correcte. Remarquons que nous utilisons en pratique la méthode de mise à jour rapide (*early update*) et que nous utilisons l'algorithme du perceptron moyenné dont l'estimation finale des poids est la moyenne de l'ensemble des vecteurs de poids générés en cours d'apprentissage (Collins, 2002).

Afin de comparer les différents modes de représentations, on utilise deux types de modèles structurés qui diffèrent par la structure du vecteur $\mathbf{\Phi}(C)$. D'une part, on définit un **modèle global creux** où $\mathbf{\Phi}(C)$ est un vecteur creux de très haute dimensionnalité valué par des fonctions indicatrices. Ces fonctions correspondent en pratique à celles des modèles baseline (**perceptron-base**) et morphologique (**perceptron+morph**) décrits dans Crabbé (2015). D'autre part, on définit un modèle, nommé GLOBAL-PROFOND, identique à celui proposé par Weiss *et al.* (2015) pour l'analyse en dépendances. Dans ce modèle, $\mathbf{\Phi}(C)$ est la concaténation des couches cachées

non linéaires $\mathbf{h}^{(i)}$ ($i > 0$) et de la couche de sortie $\mathbf{h}^{(s)}$ préalablement calculées par un modèle profond (PROFOND-1, PROFOND-2, PROFOND-1+M ou PROFOND-2+M). Le vecteur obtenu a par exemple la forme $\mathbf{\Phi}(C) = [\mathbf{h}^{(1)}; \mathbf{h}^{(2)}; \mathbf{h}^{(s)}]$. Sa dimensionalité est basse. Elle dépend du nombre et de la taille des couches cachées du modèle dense profond. Ainsi, on peut voir le modèle local sous-jacent comme un extracteur de représentations.

Ce modèle global dense peut également être vu comme un réseau de neurones similaire au réseau profond, auquel on ajouterait une couche de sortie de type perceptron entièrement connectée à la couche softmax et aux couches cachées (Weiss *et al.*, 2015).

L'estimation des paramètres du modèle global dense consiste à estimer uniquement les poids de la couche finale de type perceptron structuré. Pour les couches inférieures, nous utilisons une version pré-entraînée localement d'un modèle profond. Cette stratégie d'entraînement par étapes successives permet de gagner en efficacité.

6. Expériences

Les expériences que nous avons menées ont d'abord eu pour but d'évaluer les contributions respectives des représentations denses et des couches cachées non linéaires, ainsi que l'extension du modèle neuronal à la prédiction structurée, pour l'analyse en constituants. Nous avons également testé l'apport des informations morphologiques à ces modèles, et nous les avons comparés à des modèles creux standard comme le perceptron et le perceptron structuré. Dans cette section, nous commençons par décrire le protocole expérimental et les expériences réalisées, puis nous analysons les résultats obtenus.

6.1. *Protocole expérimental*

Pour le français, les expériences s'appuient sur les données *French Treebank* (Abeillé *et al.*, 2003) dans leur instanciation SPMRL (Seddah *et al.*, 2013). Pour l'anglais, nous utilisons le *Penn Treebank* (Marcus *et al.*, 1993). Dans toutes les expériences, les étiquettes morphosyntaxiques sont prédites à l'aide de MARMOT (Mueller *et al.*, 2013). Pour le français, cet étiqueteur prédit également les informations morphologiques (attributs GENRE, NOMBRE, MODE et MWE) pour chaque token. Les corpus arborés ont été transformés en suivant la procédure décrite en section 3.1. Suivant la procédure standard, l'évaluation est réalisée sur les arbres débinarisés à l'aide du logiciel EVALB[4], elle ignore la ponctuation.

4. `http://nlp.cs.nyu.edu/evalb/`. Pour l'anglais on utilise les paramètres de Collins. Pour le français on utilise par défaut une transposition des paramètres de Collins au *French Treebank*, ce qui revient à ignorer la ponctuation pour le calcul du F-Score. Nous utilisons également le paramétrage SPMRL décrit dans Seddah *et al.* (2013) pour les tests finaux.

Modèle de base					
$s_0.c_t$	$s_0.w_t.w$	$s_0.w_t.tag$	$s_1.c_t$	$s_1.w_t.w$	$s_1.w_t.tag$
$s_0.c_l$	$s_0.w_l.w$	$s_0.w_l.tag$	$s_1.c_l$	$s_1.w_l.w$	$s_1.w_l.tag$
$s_0.c_r$	$s_0.w_r.w$	$s_0.w_r.tag$	$s_1.c_r$	$s_1.w_r.w$	$s_1.w_r.tag$
$s_2.c_t$	$s_2.w_t.w$	$s_2.w_t.tag$	$q_0.w$	$q_1.w$	$q_2.w$
$q_3.w$	$q_0.tag$	$q_1.tag$	$q_2.tag$	$q_3.tag$	
Informations morphologiques additionnelles					
$s_0.w_t.mwe$	$s_0.w_l.mwe$	$s_0.w_r.mwe$	$s_1.w_t.mwe$	$s_0.w_t.gen$	$s_0.w_l.gen$
$q_0.w.mwe$	$q_1.w.mwe$	$q_2.w.mwe$	$q_3.w.mwe$	$s_1.w_t.gen$	$s_0.w_t.mood$
$s_0.w_t.num$	$s_0.w_l.num$	$s_1.w_t.num$	$s_0.w_l.mood$	$s_1.w_t.mood$	

Tableau 1. *Spécifications des informations utilisées*

Les symboles utilisés par les modèles neuronaux sans morphologie pour valuer le vecteur d'entrée $x_1, x_2, \ldots, x_n$ sont présentés dans la partie supérieure du tableau 1. Les notations utilisées dans ce tableau sont les mêmes que celles présentées dans la partie 4.1.1. Les modèles PROFOND-1+M et PROFOND-2+M utilisent en plus l'ensemble des attributs de la partie inférieure de ce tableau. Les patrons de traits utilisés par les perceptrons creux sont, quant à eux, décrits dans Crabbé (2015) et correspondent aux modèle baseline pour le français et l'anglais (PERCEPTRON+I et GLOBAL-PERCEPTRON+I) et au modèle additionnel capturant la morphologie pour le français (GLOBAL-PERCEPTRON+I+M).

Les hyperparamètres du réseau sont instanciés comme suit. Nous avons utilisé ReLU (pour *Rectified Linear Unit*) comme fonction d'activation pour les modèles profonds. Cette fonction se définit par $x \mapsto \max(0, x)$. Pour le français, les plongements lexicaux et les plongements de catégories syntaxiques (syntagme ou partie du discours) ont respectivement 64 et 32 coefficients pour les modèles DENSE LIN et 32 et 16 pour les modèles profonds. Pour l'anglais, ils ont respectivement 64 et 32 coefficients. Le nombre d'itérations est fixé à 12 pour le modèle RLM et les modèles profonds, 8 pour le modèle DENSE LIN. Pour chaque modèle nous avons fait varier le pas d'apprentissage (0,01 et 0,02), la constante α (10^{-6} et 10^{-7}) et le nombre d'unités dans les couches cachées (128, 256, 512). Pour chaque modèle, nous présentons l'entraînement qui a donné le meilleur résultat sur le corpus de développement. Notons que pour les modèles profonds, de manière systématique, les plus grandes couches cachées ont donné les meilleurs résultats.[5] Pour les modèles neuronaux utilisant des informations morphologiques, les attributs GENRE, NOMBRE, MODE et MWE ont respectivement des plongements à 1, 1, 4 et 16 coefficients. Tous les paramètres, y compris les plongements, sont initialisés aléatoirement.

5. Nous n'avons pas testé de couches cachées de dimensions supérieures à 512 car le temps d'entraînement devient important.

6.2. *Résultats et discussions*

Nous présentons dans le tableau 2 les résultats pour le français sur le corpus de développement pour les différents modèles introduits plus haut, et dans le tableau 3 les résultats sur l'anglais pour les modèles profonds.

	Taille du faisceau				
Entraînement local	1	2	4	8	16
RLM	78,56	78,08	78,49	78,56	78,55
DENSE LIN	78,43	79,19	79,38	79,44	79,51
PROFOND-1	81,75	82,09	82,13	82,24	82,21
PROFOND-2	82,31	82,67	82,73	82,76	82,74
PROFOND-1+M	82,44	82,95	83,05	83,12	83,09
PROFOND-2+M	**82,75**	82,95	82,94	82,97	82,98
PERCEPTRON+I	79,77	77,21	75,16	74,16	73,27
Entraînement global	1	2	4	8	16
GLOBAL-PROFOND	82,00	82,78	83,47	83,61	83,46
GLOBAL-PROFOND+M	82,22	82,99	83,34	83,42	83,50
GLOBAL-PERCEPTRON+I	79,53	81,33	82,22	82,79	82,55
GLOBAL-PERCEPTRON+I+M	79,96	82,20	82,90	83,25	83,68

Tableau 2. *Résultats pour le français sur le corpus de développement*

	Taille du faisceau				
Entraînement local	1	2	4	8	16
PROFOND-1	88,97	89,53	89,62	89,64	89,65
PROFOND-2	89,46	89,73	89,79	89,84	89,85
PERCEPTRON+I	86,74	83,59	81,80	80,76	80,19
Entraînement global	1	2	4	8	16
GLOBAL-PROFOND	89,26	90,08	90,62	90,64	90,85
GLOBAL-PERCEPTRON+I	87,15	89,05	89,67	89,93	90,15

Tableau 3. *Résultats pour l'anglais sur le corpus de développement*

6.2.1. *Contribution des représentations denses et des interactions*

Dans le tableau 2, nous observons que l'introduction des plongements apporte une amélioration sur le modèle de base. Toutefois, le réseau dense linéaire en décodage glouton (taille de faisceau de 1) reste en deçà d'un perceptron qui prend en compte des interactions entre variables. Par ailleurs, le passage aux modèles profonds améliore considérablement l'analyse syntaxique. Cela suggère qu'il est crucial de prendre

en compte les interactions entre variables, indépendamment de l'utilisation de représentations creuses ou denses.

Des travaux précédents ont étudié l'utilisation de plongements pré-entraînés pour l'analyse en constituants (Andreas et Klein, 2014). L'amélioration très faible avait mené à la conclusion que les informations données par les plongements étaient redondantes avec des informations déjà accessibles comme les catégories morphosyntaxiques. Nos expériences complètent cette conclusion en mettant en valeur l'importance des interactions. En effet, dans le tableau 2, le modèle PROFOND-1 apporte une amélioration d'environ trois points de F-mesure par rapport au modèle DENSE LIN.

6.2.2. *Effets combinés de l'apprentissage global et de la recherche par faisceau*

On observe dans le tableau 2 que le perceptron local (PERCEPTRON+I) se comporte très mal quand on utilise une recherche par faisceau. Les résultats se dégradent avec la taille du faisceau. Ce résultat concorde avec l'étude sur l'analyse en dépendances de Zhang et Nivre (2012) où il est montré que la recherche par faisceau n'est efficace que lorsqu'on utilise un modèle d'apprentissage global avec la même taille de faisceau que lors de l'entraînement. En revanche, pour les modèles neuronaux locaux où les scores des actions sont normalisés (probabilités), on remarque au contraire de légères améliorations qui plafonnent avec une petite taille de faisceau (tableau 2). Cela suggère que la normalisation des scores réduit l'effet de biais locaux.

6.2.3. *Extension à l'apprentissage structuré*

À taille de faisceau égale, le modèle structuré que nous avons présenté en section 5 apporte une amélioration sur le modèle local qu'il utilise. Cette amélioration est surtout visible pour le modèle sans morphologie, et pour une taille de faisceau supérieure ou égale à 4. Notons que le modèle GLOBAL-PROFOND dépend du modèle local pré-entraîné sur lequel il se base. Notamment la dimensionnalité des représentations utilisées par le perceptron dépend de celle des couches cachées du modèle local. Pour tester ces effets, nous avons entraîné des modèles GLOBAL-PROFOND à partir de modèles locaux (avec morphologie) en faisant varier la taille des couches cachées. On présente en figure 5 le nombre de mises à jour du vecteur de poids du perceptron en fonction du rang des itérations pour différentes dimensionnalités des couches cachées. Ces résultats sont comparés avec ceux d'un perceptron structuré standard (noté « perceptron » dans la figure 5). On observe que lorsque la dimensionnalité des représentations est plus élevée, la courbe converge vers de plus petites valeurs. Cela est également vrai pour le modèle creux de très haute dimensionnalité qui manifeste une courbe plus prototypique. Des représentations de trop faible dimensionnalité ne donnent plus assez d'indices au perceptron pour discriminer les données. Cela expliquerait que les modèles utilisant les informations morphologiques ne profitent pas vraiment du modèle global.

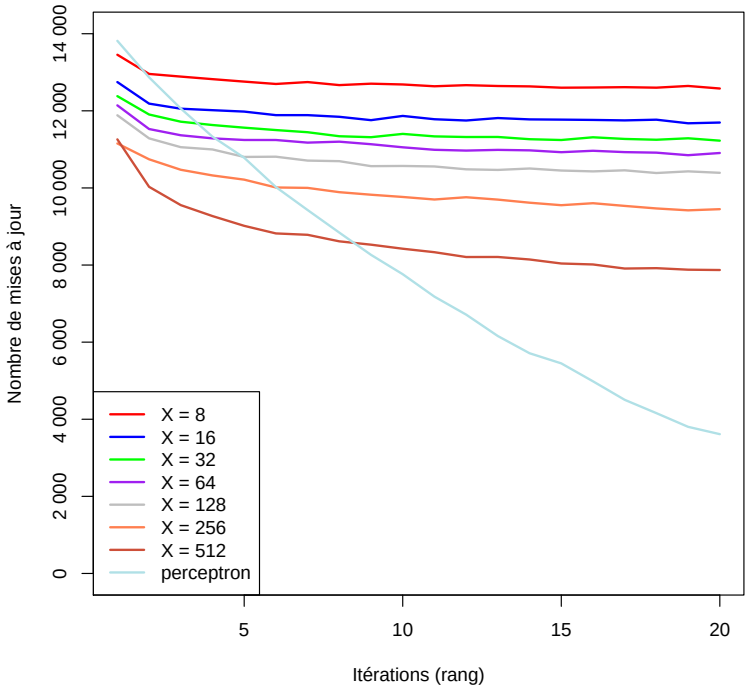

Figure 5. *Nombre de mises à jour des poids par le perceptron structuré en fonction du nombre d'itérations sur le corpus et de la taille X des couches cachées du modèle local pré-entraîné.*

6.2.4. *Informations morphologiques*

De manière systématique, l'ajout d'informations morphologiques permet d'améliorer l'analyse syntaxique. Une exception à cela est le modèle neuronal structuré. Notons que l'interface que nous avons utilisée pour les attributs morphologiques est très simple et peut s'étendre facilement à un nombre arbitraire d'attributs. Nous prévoyons de la tester sur des langues à morphologie plus riche que le français, et d'évaluer d'autres façons de prendre en compte les informations morphologiques.

6.2.5. *Discussion*

Nous présentons en tableau 4 les résultats sur les corpus de test pour l'anglais et le français, ainsi que quelques résultats publiés dans la littérature. En particulier, Durrett et Klein (2015) présentent un CRF (Lafferty *et al.*, 2001) dont les représentations sont construites par un réseau de neurones. Les autres modèles utilisent des grammaires à

annotations latentes, soit des PCFG-LA (Petrov *et al.*, 2006 ; Le Roux *et al.*, 2014), soit des grammaires de substitution d'arbres (Shindo *et al.*, 2012). Le Roux *et al.* (2014) proposent d'utiliser la décomposition duale pour combiner la sortie de plusieurs systèmes prédisant des catégories morphosyntaxiques, une reconnaissance des expressions polylexicales et des arbres en constituants issus de différentes grammaires PCFG-LA. Tous ces modèles utilisent des systèmes d'inférence basés sur la programmation dynamique et pour certains, la combinaison de résultats de différents analyseurs (Shindo *et al.*, 2012 ; Le Roux *et al.*, 2014).

À l'heure actuelle, les meilleurs systèmes pour l'analyse en constituants sont issus de méthodes de combinaison d'analyseurs dans le paradigme non lexicalisé (Le Roux *et al.*, 2014) ou de méthodes incorporant de la semi-supervision (Durrett et Klein, 2015). Comme illustré en table 2, l'approche lexicalisée gloutonne avec *deep learning* élémentaire obtient des résultats notables, malgré la simplicité de son système d'inférence. En effet, dans des conditions d'évaluation comparables, en excluant les méthodes à réordonnancement ou d'ensemble d'analyseurs, le réseau profond glouton est compétitif avec le modèle de Petrov *et al.* (2006) ou le perceptron structuré sur le français.

La perspective principale concernant l'approche *deep learning* pour l'analyse syntaxique lexicalisée est de tirer parti des représentations vectorielles pour acquérir et améliorer les représentations lexicales notamment à partir de sources externes aux seuls corpus arborés habituellement utilisés pour entraîner les analyseurs syntaxiques. Mais on peut espérer à plus long terme une amélioration substantielle du comportement de l'analyseur par l'acquisition de « plongements bilexicaux ».

7. Perspectives et conclusion

Nous avons présenté différents modèles de pondération de type réseaux de neurones pour l'analyse en constituants par transitions. Nous avons confronté ces modèles à des modèles comme le perceptron ou le perceptron structuré, pour différents scénarios expérimentaux. Nous avons d'abord observé que l'analyseur lexicalisé à pondération *deep learning* se comporte étonnamment bien en utilisant une recherche gloutonne, et ce, contrairement aux observations initiales de Sagae et Lavie (2006) qui se fondaient sur un modèle creux. En revanche, dans le cas d'une inférence en faisceau, le modèle permet au mieux de répliquer dans des conditions d'évaluation similaires les résultats de l'état de l'art. On peut toutefois espérer une progression significative en travaillant davantage la représentation de l'historique de la dérivation et l'acquisition de relations bilexicales sur de grandes masses de données.

Deux perspectives à plus court terme nous permettraient d'améliorer l'analyseur glouton que nous avons présenté. Premièrement, l'utilisation d'oracles dynamiques (Goldberg et Nivre, 2012) devrait permettre d'améliorer la recherche en laissant l'analyseur explorer une plus grande partie de l'espace de recherche lors de l'entraînement. Deuxièmement, nous n'avons pas de mécanisme dédié au traitement des mots incon-

		Français		Anglais
Système	Décodage	F-Score	F-Score SPMRL	F-Score
Analyseurs simples				
PERCEPTRON+I	glouton	78,99	77,27	86,5
PROFOND-2+M	glouton	82,63	80,12	89,0
PROFOND-2+M	faisceau = 8	82,98	80,46	89,4
GLOBAL-PERCEPTRON+I	faisceau = 8	82,83	80,13	90,0
GLOBAL-PROFOND	faisceau = 8	83,03	80,52	90,1
Berkeley (Petrov *et al.*, 2006)	CKY	82,24	79,99	90,1
Durrett et Klein (2015)†	CKY	-	81,25	91,1
Hall *et al.* (2014)	CKY	-	79,70	89,2
Shindo *et al.* (2012)	CKY	-	-	91,1
Conditions étendues				
Shindo *et al.* (2012) ensemble	CKY	-	-	92,4
Le Roux *et al.* (2014) ensemble	CKY	-	83,80	-

Tableau 4. *Résultats finaux pour le français et l'anglais sur le corpus de test. Pour l'anglais, aucun modèle n'utilise d'informations morphologiques.* † *: modèle semi-supervisé (pré-entraînement des plongements lexicaux sur des données additionnelles non annotées).*

nus. L'utilisation de méthodes qui exploitent des similarités orthographiques entre symboles pour extraire des informations à partir des séquences de caractères semble particulièrement prometteuse (Santos et Zadrozny, 2014 ; Ballesteros *et al.*, 2015).

8. Bibliographie

Abeillé A., Clément L., Toussenel F., « Building a Treebank for French », *Treebanks : Building and Using Parsed Corpora*, Springer, p. 165-188, 2003.

Andreas J., Klein D., « How much do word embeddings encode about syntax ? », *Proceedings of ACL*, 2014.

Ballesteros M., Dyer C., Smith N. A., « Improved Transition-based Parsing by Modeling Characters instead of Words with LSTMs », *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Lisbon, Portugal, p. 349-359, September, 2015.

Bengio Y., Ducharme R., Vincent P., « A Neural probabilistic language model », *Journal of Machine Learning Research*, vol. 3, p. 1137-1155, 2003.

Charniak E., « A Maximum-entropy-inspired Parser », *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*, NAACL 2000, Association for Computational Linguistics, Stroudsburg, PA, USA, p. 132-139, 2000.

Chen D., Manning C. D., « A Fast and Accurate Dependency Parser using Neural Networks », *Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

Coavoux M., Crabbé B., « Comparaison d'architectures neuronales pour l'analyse syntaxique en constituants », *Actes de la 22e conférence sur le Traitement Automatique des Langues Naturelles*, Association pour le Traitement Automatique des Langues, Caen, France, p. 293-304, June, 2015.

Collins M., « Three Generative, Lexicalised Models for Statistical Parsing », *Proceedings of the Eighth Conference on European Chapter of the Association for Computational Linguistics*, EACL '97, Association for Computational Linguistics, Stroudsburg, PA, USA, p. 16-23, 1997.

Collins M., « Discriminative Training Methods for Hidden Markov Models : Theory and Experiments with Perceptron Algorithms », *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, Association for Computational Linguistics, Stroudsburg, PA, USA, p. 1-8, 2002.

Collobert R., Weston J., « A Unified Architecture for Natural Language Processing : Deep Neural Networks with Multitask Learning », *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, ACM, New York, NY, USA, p. 160-167, 2008.

Crabbé B., « An LR-inspired generalized lexicalized phrase structure parser », *Proceedings of the twenty-fifth International Conference on Computational Linguistics*, Dublin, Ireland, August, 2014.

Crabbé B., « Multilingual discriminative lexicalized phrase structure parsing », *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Lisbon, Portugal, p. 1847-1856, September, 2015.

Duchi J., Hazan E., Singer Y., « Adaptive Subgradient Methods for Online Learning and Stochastic Optimization », *J. Mach. Learn. Res.*, vol. 12, p. 2121-2159, July, 2011.

Durrett G., Klein D., « Neural CRF Parsing », *Proceedings of the Association for Computational Linguistics*, Association for Computational Linguistics, Beijing, China, July, 2015.

Dyer C., Ballesteros M., Ling W., Matthews A., Smith N. A., « Transition-Based Dependency Parsing with Stack Long Short-Term Memory », *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1 : Long Papers)*, Association for Computational Linguistics, Beijing, China, p. 334-343, July, 2015.

Goldberg Y., Nivre J., « A Dynamic Oracle for Arc-Eager Dependency Parsing », *Proceedings of COLING 2012*, The COLING 2012 Organizing Committee, Mumbai, India, p. 959-976, December, 2012.

Hall D., Durrett G., Klein D., « Less Grammar, More Features », *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, Association for Computational Linguistics, Baltimore, Maryland, June, 2014.

Henderson J., « Inducing History Representations for Broad Coverage Statistical Parsing », *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, Association for Computational Linguistics, Stroudsburg, PA, USA, p. 24-31, 2003.

Hinton G. E., Osindero S., Welling M., Teh Y. W., « Unsupervised Discovery of Nonlinear Structure Using Contrastive Backpropagation. », *Cognitive Science*, vol. 30, n° 4, p. 725-731, 2006.

Klein D., Manning C. D., « Accurate Unlexicalized Parsing », *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, Association for Computational Linguistics, Stroudsburg, PA, USA, p. 423-430, 2003.

Koo T., Carreras X., Collins M., « Simple Semi-supervised Dependency Parsing », *Proceedings of ACL-08 : HLT*, Association for Computational Linguistics, Columbus, Ohio, p. 595-603, June, 2008.

Lafferty J. D., McCallum A., Pereira F. C. N., « Conditional Random Fields : Probabilistic Models for Segmenting and Labeling Sequence Data », *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, p. 282-289, 2001.

Le Roux J., Rozenknop A., Constant M., « Syntactic Parsing and Compound Recognition via Dual Decomposition : Application to French », *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics : Technical Papers*, Dublin City University and Association for Computational Linguistics, Dublin, Ireland, p. 1875-1885, August, 2014.

Marcus M. P., Santorini B., Marcinkiewicz M. A., « Building a Large Annotated Corpus of English : The Penn Treebank », *Computational Linguistics*, vol. 19, n° 2, p. 313-330, 1993.

Mikolov T., Chen K., Corrado G., Dean J., « Efficient Estimation of Word Representations in Vector Space », *CoRR*, 2013.

Montufar G. F., Pascanu R., Cho K., Bengio Y., « On the number of linear regions of deep neural networks », *Advances in neural information processing systems*, 2014.

Mueller T., Schmid H., Schütze H., « Efficient Higher-Order CRFs for Morphological Tagging », *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Seattle, Washington, USA, p. 322-332, October, 2013.

Nederhof M.-J., « Weighted deductive parsing and Knuth's algorithm », *Computational Linguistics*, 2003.

Petrov S., Barrett L., Thibaux R., Klein D., « Learning Accurate, Compact, and Interpretable Tree Annotation », *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, Sydney, Australia, p. 433-440, July, 2006.

Polyak B. T., Juditsky A. B., « Acceleration of Stochastic Approximation by Averaging », *SIAM J. Control Optim.*, vol. 30, n° 4, p. 838-855, July, 1992.

Sagae K., Lavie A., « A best-first probabilistic shift-reduce parser », *Proceedings of the COLING/ACL on Main conference poster sessions*, Association for Computational Linguistics, p. 691-698, 2006.

Santos C. D., Zadrozny B., « Learning Character-level Representations for Part-of-Speech Tagging », *in* T. Jebara, E. P. Xing (eds), *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, JMLR Workshop and Conference Proceedings, p. 1818-1826, 2014.

Seddah D., Tsarfaty R., Kübler S., Candito M., Choi J. D., Farkas R., Foster J., Goenaga I., Gojenola Galletebeitia K., Goldberg Y., Green S., Habash N., Kuhlmann M., Maier W., Nivre J., Przepiórkowski A., Roth R., Seeker W., Versley Y., Vincze V., Woliński M., Wróblewska A., de la Clergerie E. V., « Overview of the SPMRL 2013 Shared Task : A Cross-Framework Evaluation of Parsing Morphologically Rich Languages », *Proceedings of the*

Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages, Association for Computational Linguistics, Seattle, Washington, USA, p. 146-182, October, 2013.

Shindo H., Miyao Y., Fujino A., Nagata M., « Bayesian Symbol-Refined Tree Substitution Grammars for Syntactic Parsing », *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, Association for Computational Linguistics, Jeju Island, Korea, p. 440-448, July, 2012.

Watanabe T., Sumita E., « Transition-based Neural Constituent Parsing », *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1 : Long Papers)*, Association for Computational Linguistics, Beijing, China, p. 1169-1179, July, 2015.

Weiss D., Alberti C., Collins M., Petrov S., « Structured Training for Neural Network Transition-Based Parsing », *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1 : Long Papers)*, Association for Computational Linguistics, Beijing, China, p. 323-333, July, 2015.

Zhang Y., Clark S., « Transition-based Parsing of the Chinese Treebank Using a Global Discriminative Model », *Proceedings of the 11th International Conference on Parsing Technologies*, IWPT '09, Association for Computational Linguistics, Stroudsburg, PA, USA, p. 162-171, 2009.

Zhang Y., Nivre J., « Analyzing the Effect of Global Learning and Beam-Search on Transition-Based Dependency Parsing. », *COLING (Posters)*, p. 1391-1400, 2012.

Zhu M., Zhang Y., Chen W., Zhang M., Zhu J., « Fast and Accurate Shift-Reduce Constituent Parsing », *ACL (1)*, The Association for Computer Linguistics, p. 434-443, 2013.