Ne nous arrêtons pas en si bon chemin : améliorations de l'apprentissage global d'analyseurs en dépendances par transition

Lauriane Aufrant[1, 2] Guillaume Wisniewski[1] François Yvon[1]
(1) LIMSI, CNRS, Univ. Paris-Sud, Université Paris-Saclay, 91 405 Orsay, France
(2) DGA, 60 boulevard du Général Martial Valin, 75 509 Paris, France
{lauriane.aufrant, guillaume.wisniewski,francois.yvon}@limsi.fr

RÉSUMÉ

Dans cet article, nous proposons trois améliorations simples pour l'apprentissage global d'analyseurs en dépendances par transition de type ARCEAGER : un oracle non déterministe, la reprise sur le même exemple après une mise à jour et l'entraînement en configurations sous-optimales. Leur combinaison apporte un gain moyen de 0,2 UAS sur le corpus SPMRL. Nous introduisons également un cadre général permettant la comparaison systématique de ces stratégies et de la plupart des variantes connues. Nous montrons que la littérature n'a étudié que quelques stratégies parmi les nombreuses variations possibles, négligeant ainsi plusieurs pistes d'améliorations potentielles.

ABSTRACT

Don't Stop Me Now ! Improved Update Strategies for Global Training of Transition-Based Dependency Parsers.

In this paper, we propose three simple improvements for global training of ARCEAGER transition-based dependency parsers : a non-deterministic oracle, resuming on the same example after an update and training in suboptimal configurations. Their combination yields an average improvement of 0.2 UAS on the SPMRL dataset. Moreover, we introduce a general framework that enables systematic comparison of those strategies and most variants. We show that the literature has only explored a small number of strategies among the multiple design variants, neglecting potential improvements.

MOTS-CLÉS : Analyse en dépendances, Analyse par transition, Oracle.

KEYWORDS: Dependency parsing, Transition-based, Oracle.

Les principaux algorithmes d'analyse en dépendances se répartissent en deux catégories, les analyseurs à base de graphes et par transition [1]. Nous nous intéressons ici à la deuxième catégorie, et plus particulièrement aux méthodes incrémentales avec recherche inexacte. Malgré l'efficacité de leur entraînement et la qualité de leurs prédictions, nous affirmons que ces algorithmes disposent encore d'une marge d'amélioration en précision, accessible par des moyens simples et sans surcoût de calcul.

Après un rappel de l'état de l'art (§ 1) et l'étude de ses limites, nous proposons et validons expérimentalement (§ 2 et § 3) trois améliorations simples de l'apprentissage. Motivés par la possibilité de rapprochement de ces stratégies avec d'autres méthodes classiques, nous introduisons (§ 4) un formalisme permettant une description unifiée de ces analyseurs, ouvrant la voie à d'autres améliorations.

1. Respectivement *graph-based* et *transition-based* dans la littérature anglo-saxonne.

1 Principe d'un analyseur syntaxique par transition

Nous rappelons, dans cette section, les principes de fonctionnement des analyseurs syntaxiques par transition ainsi que les méthodes d'apprentissage associées. Par souci de clarté, nous ne considérerons ici que les analyseurs ARCEAGER (Nivre, 2008). Notre analyse se généralise facilement aux autres analyseurs par transition en dépendances et en constituants (Coavoux & Crabbé, 2015).

1.1 Prédiction structurée d'un arbre de dépendance

Dans un système par transition, la sortie est construite de manière incrémentale : dans une configuration c décrivant l'analyse partielle d'une phrase, l'analyseur choisira une transition t ; ce choix lui permettra d'étendre la sortie en cours de construction et de passer dans une configuration $c' = c \circ t$ qualifiée de « successeur » de c, qui est alors un antécédent de c'. L'analyse d'une phrase se déduit alors d'une *dérivation*, c'est-à-dire d'une suite de transitions permettant de transformer une configuration initiale décrivant un arbre de dépendance vide en une configuration finale dans laquelle chaque mot est relié à son gouverneur. La Figure 1 illustre un exemple de dérivation.

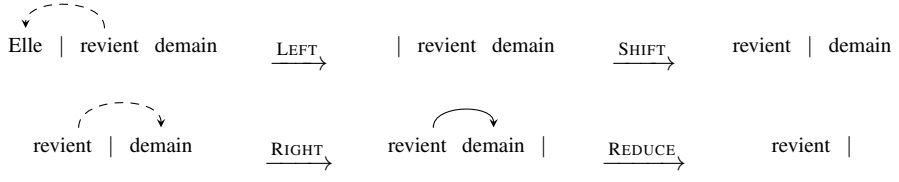

FIGURE 1: Exemple de dérivation ARCEAGER, montrant la configuration courante (pile et buffer de part et d'autre du séparateur) et son évolution après chacune des quatre transitions ARCEAGER.

À chaque étape de l'analyse, le score de chaque transition est calculé, généralement par un modèle linéaire réalisant pour cette transition le produit scalaire entre le vecteur ϕ de caractéristiques de la configuration courante et un vecteur de paramètres $\mathbf{w}$. Le score (global) d'un arbre est défini comme la somme des scores des transitions le construisant. Analyser syntaxiquement une phrase consiste à trouver l'arbre de plus grand score. Cette recherche est généralement faite, pour des raison de complexité, par des méthodes approchées comme la recherche gloutonne (Nivre, 2008) ou la recherche en faisceau *(beam search)* (Zhang & Clark, 2008), qui ont un temps d'inférence linéaire par rapport à la taille de la phrase.

Dans la suite de ce travail, nous nous concentrerons sur les analyseurs en faisceau qui constituent aujourd'hui l'état de l'art en analyse syntaxique : le faisceau contient initialement uniquement la configuration initiale ; puis, à chaque étape (ou *profondeur*) de la dérivation, l'ensemble des successeurs des configurations du faisceau est calculé, puis tronqué aux k configurations de plus grand score. Le décodage se poursuit jusqu'à ce qu'une configuration finale soit atteinte.

Le choix d'utiliser une méthode d'inférence incrémentale soulève deux questions. Premièrement, elle entraîne une intrication très forte entre les exemples utilisés pour estimer le vecteur de paramètres $\mathbf{w}$ et les exemples qui devront être analysés en test : toute modification du vecteur de paramètres, au cours de l'apprentissage, modifiera directement les configurations qui seront explorées, pour les exemples qui suivent, par l'algorithme de recherche approché. Les configurations visitées en

apprentissage doivent donc être engendrées de manière à assurer leur similarité aux configurations qui seront rencontrées en test. Notons que les exemples ne sont pas *i.i.d.*, ce qui remet en cause l'une des hypothèses fondamentales de l'apprentissage statistique.

Deuxièmement, tout arbre de dépendance, et notamment l'arbre de référence, peut correspondre à plusieurs dérivations. Cette ambiguïté, qualifiée de « fallacieuse » *(spurious ambiguity)*, complique l'estimation des paramètres, puisque la plupart des algorithmes d'apprentissage exploitent une *unique référence* qui indique la « direction » de la mise à jour (par exemple dans le calcul du gradient).

Si toutes les méthodes incrémentales souffrent de ces deux travers (Knyazeva *et al.*, 2015), les analyseurs syntaxiques soulèvent un troisième problème : les distributions de caractéristiques proposées au classifieur en apprentissage sont différentes en début et fin de dérivation, d'une part parce que ces caractéristiques correspondent à des *tokens* de début (avec majuscule, pronoms personnels des langues SVO et SOV) ou de fin de phrase (ponctuations, verbes des langues SOV), d'autre part du fait de la proximité de ces *tokens* aux feuilles (déterminants) ou à la racine (verbes) des arbres de dépendance. En effet, les systèmes de type décalage-réduction *(shift-reduce)* traitent, en fin de dérivation, davantage de gros sous-arbres, réduits à leur racine, puisque les dernières transitions correspondent à l'analyse des *tokens* finaux et au rattachement des « grandes structures » de la phrase.

1.2 Apprentissage des analyseurs en faisceau

Cadre général L'estimation d'un vecteur de paramètres permettant de guider une recherche en faisceau est un problème au cœur de nombreux domaines et les solutions proposées varient fortement d'une communauté à l'autre. Par exemple, en traduction automatique, les paramètres de la fonction de score sont appris en assurant que les n meilleures traductions d'une phrase sont bien ordonnées selon une mesure de qualité (Och & Ney, 2002; Hopkins & May, 2011) ; de manière similaire, Xu *et al.* (2009) proposent d'utiliser un critère de *ranking* pour résoudre des problèmes de planification.

En ce qui concerne l'apprentissage des analyseurs syntaxiques, les travaux de Collins & Roark (2004) ont introduit une approche à base de perceptron moyenné pour estimer le vecteur de paramètres permettant de calculer le score d'une transition. Cette approche a été adoptée dans la quasi-totalité des travaux ultérieurs [2]. L'apprentissage se déroule en-ligne (*online*) : pour chaque phrase de l'ensemble d'apprentissage, l'analyseur commence par prédire l'arbre de dépendance en utilisant les poids courants du modèle dans sa recherche en faisceau ; dès qu'une erreur est détectée, le décodage est interrompu et le vecteur de paramètres est mis à jour en pénalisant une dérivation « négative » se terminant dans la configuration c^- et en renforçant une dérivation « positive » se terminant dans une configuration c^+. (Zhang & Nivre, 2012) montre expérimentalement que dans le cas d'un analyseur en faisceau, il est nécessaire d'utiliser un critère d'*apprentissage global* (un perceptron structuré (Collins, 2002)) et de considérer dans les mises à jour l'ensemble des caractéristiques observées lors d'une dérivation et non pas celles d'une seule transition. La définition de l'algorithme d'apprentissage revient donc, essentiellement, à spécifier c^-, c^+, ainsi que la manière de détecter une erreur de décodage. Plusieurs stratégies ont été proposées, nous présentons dans le paragraphe suivant les deux principales, EARLY UPDATE et MAX VIOLATION.

Le fait d'imbriquer les étapes d'apprentissage et d'inférence a globalement pour effet [3] de rendre

2. Des approches comme (Coavoux & Crabbé, 2015) estiment ces paramètres avec un réseau de neurones. Notre analyse reste valable car leur apprentissage nécessite de déterminer une unique référence et porte sur le score de la meilleure solution.

3. Cette imbrication n'a été, pour le moment, justifiée que de manière intuitive et empirique, même si son principe est au

les configurations vues en apprentissage « similaires » à celles du test : l'apprentissage visite, en effet, des configurations engendrées par la même fonction de décision (c.-à-d. par le même vecteur de paramètres) que celle qui sera utilisée pour analyser les phrases du test. On peut ainsi espérer que les erreurs faites lors de la prédiction des transitions seront similaires lors de l'apprentissage et du test et que l'analyseur apprendra ainsi à limiter la propagation des erreurs.

Stratégies EARLY UPDATE et MAX VIOLATION Dans sa version standard, l'algorithme du perceptron fait évoluer les paramètres en comparant la dérivation complète prédite avec une dérivation complète de référence, précalculée à l'aide de règles *ad hoc*[4]. Par opposition, le principe EARLY UPDATE (Collins & Roark, 2004) conduit à interrompre le décodage et à réaliser une mise à jour dès que cette dérivation de référence sort du faisceau. Les erreurs ne faisant que s'accumuler, il est en effet certain dès cette étape que la prédiction finale sera erronée et qu'une mise à jour sera nécessaire ; ne pas attendre la fin de la dérivation permet d'éviter que des configurations résultant d'une accumulation d'erreurs et donc très éloignées de celles qui seront visitées en test n'affectent la mise à jour. Les paramètres évoluent alors pour renforcer les caractéristiques de c^+ (la référence sortie du faisceau) et inversement pour pénaliser celles de c^- (la meilleure hypothèse courante du faisceau). La fin de la dérivation est ignorée et l'apprentissage continue avec une autre phrase.

L'approche MAX VIOLATION (Huang *et al.*, 2012) s'appuie sur des principes similaires : lorsque la référence sort du faisceau, on recherche la profondeur de dérivation maximisant l'écart entre le score de la référence et celui de la meilleure hypothèse courante du faisceau. La mise à jour implique alors l'hypothèse courante à cette profondeur et un antécédent de la configuration finale de référence. Cette méthode permet de couvrir un plus grand nombre de transitions que la méthode précédente, tout en garantissant que les mises à jour sont déclenchées par des erreurs de score global *(violations)* ; elle permet de garantir théoriquement, comme EARLY UPDATE, la convergence de l'apprentissage tout en étant, expérimentalement, plus rapide. Choisir exactement l'écart maximal de score permet de s'assurer qu'on n'inclut les transitions dans la mise à jour que tant que celle de la branche positive est plus mal notée que la négative ; nous verrons en § 2 que cette garantie a toutefois un coût.

Perceptron moyenné À la fin de l'apprentissage, le vecteur de paramètres est défini comme la moyenne de l'ensemble des valeurs prises durant l'apprentissage. Comme le montre (Collins, 2002), considérer la moyenne du vecteur de paramètres permet d'améliorer les performances en prédiction.

Une difficulté des approches en faisceau, ignorée par la littérature, est l'interaction de ce moyennage final avec l'apprentissage global. Comme les mises à jour globales impliquent plusieurs transitions, les pas de temps d'apprentissage sont désynchronisés des pas de temps de décodage et une moyenne faite suivant l'une ou l'autre échelle ne donnera pas les mêmes modèles. Nos travaux utilisent l'échelle de temps du décodage, ce qui revient à exécuter la mise à jour successivement sur chaque transition.

2 Amélioration des stratégies d'apprentissage

Les stratégies EARLY UPDATE et MAX VIOLATION présentent plusieurs limitations :

1. Elles reposent toutes deux sur une dérivation de référence précalculée de façon déterministe. Dans le système ARCEAGER, dans lequel les arbres de référence ont souvent de nombreuses

cœur des algorithmes d'apprentissage par imitation (Ross & Bagnell, 2010). En dépit des travaux de Chang *et al.* (2015) qui explicitent ce lien, très peu de garanties théoriques ont été apportées aux algorithmes d'apprentissage que nous présentons ici.

4. Ces règles visent à résoudre le problème des références multiples, en décidant de manière arbitraire quelle transition choisir en cas d'ambiguïté.

dérivations, il arrive donc que le modèle soit mis à jour alors même que l'arbre de dépendance de référence a été prédit, mais *via* une autre dérivation.

2. Ces deux méthodes opèrent leurs mises à jour sur des préfixes des dérivations et introduisent un biais dans la distribution des caractéristiques vues à l'apprentissage : les configurations de début de dérivation sont sur-représentées par rapport à celles plus proches de la fin.
3. Le décodage est initialisé depuis une configuration vide, depuis laquelle tout arbre de référence est accessible : les mises à jour ont toujours pour effet de renforcer des configurations correspondant à des arbres de référence. Or, au moment du test, les configurations évaluées peuvent être erronées et présenter des caractéristiques jamais vues à l'entraînement. Pour rendre le modèle capable de prendre de bonnes décisions dans ces cas, il est crucial (Goldberg & Nivre, 2013) de considérer également des configurations non optimales durant l'apprentissage.

Nous allons montrer comment les stratégies EARLY UPDATE et MAX VIOLATION peuvent être étendues pour apporter des solutions à chacun de ces problèmes. Tout le code nécessaire à leur mise en place est librement disponible (Aufrant & Wisniewski, 2016).

2.1 Oracles non déterministes pour les analyseurs en faisceau

Goldberg & Nivre (2012) ont montré que, dans le cas d'un analyseur glouton, adapter c^+ à l'état dans lequel se trouve l'analyseur et non de manière arbitraire permet d'éviter les erreurs de mise à jour et d'améliorer significativement les performances. L'oracle qualifié de *non déterministe* qu'ils proposent n'a toutefois jamais été généralisé au cas d'un analyseur ARCEAGER en faisceau.

La méthode EARLY UPDATE se généralise facilement aux oracles non déterministes : il suffit de changer le critère d'erreur pour que celui-ci détecte, non pas le moment où la référence précalculée sort du faisceau, mais le moment où plus aucune hypothèse du faisceau ne peut générer l'arbre de référence. L'oracle reconnaît ainsi toutes les références concurrentes, d'où son non déterminisme.

Ce critère d'erreur est implanté en agrégeant au décodage, sur la totalité des transitions d'une dérivation, la fonction de coût de Goldberg & Nivre (2013). Cette fonction $\textsc{Cost}(t)$ compte le nombre de dépendances correctes qui ne pourront plus être prédites si l'on choisit la transition t. Formellement, une dérivation aboutissant dans une configuration c à partir de la configuration c_0 est qualifiée de *correcte* si :

$$\textsc{Correct}(c|c_0) \equiv \exists n, c = c_0 \circ t_1 \circ \ldots \circ t_n, \textsc{Cost}(t_1) = \cdots = \textsc{Cost}(t_n) = 0 \qquad (1)$$

En cas d'erreur, on choisit pour c^+ la dérivation la mieux notée (à des fins de stabilité) parmi les références qui viennent de sortir du faisceau, à savoir les hypothèses qui sont encore optimales dans le faisceau avant troncature. L'algorithme 1 formalise l'oracle non déterministe EARLY UPDATE.

Le développement d'un oracle non déterministe pour MAX VIOLATION est plus compliqué puisque cette méthode nécessite, pour calculer les écarts de scores, de déterminer une référence bien après la divergence faisceau/références et il n'existe alors plus de référence unique naturellement plus proche du faisceau. La détermination de l'écart minimal par rapport à l'ensemble (potentiellement grand) des références n'est, en général, pas calculable efficacement. Il est donc inévitable de favoriser arbitrairement une référence par rapport aux autres, mais nous réduisons toutefois cet effet.

Notre proposition pour engendrer des références qui soient cohérentes avec les prédictions actuelles de l'analyseur est la suivante : après détection d'une erreur, et en notant c_0^+ la meilleure référence qui vient de sortir du faisceau, on construit une configuration finale c_f^+ par recherche en faisceau

Algorithme 1 : Méthode de recherche de c^- et c^+ pour la configuration initiale c_0 et l'arbre de référence y dans le cas d'une stratégie EARLY UPDATE avec un oracle non déterministe.

Function EARLYUPDATEORACLE*(c_0, y,* **w***)*
- `// Faisceau initialisé avec la configuration initiale`
- $B \leftarrow \{c_0\}$
- **while** $\exists c \in B, \neg$FINAL(c) **do**
  - `// Mise à jour de` B `avec les` k `successeurs de plus grand score`
  - succ $\leftarrow \cup_{c \in B}$NEXT$(c)$
  - $B \leftarrow k$-best(succ, **w**)
  - `// vérifie s'il reste une référence dans` B
  - **if** $\forall c \in B, \neg$CORRECT$_y(c|c_0)$ **then**
    - gold $= \{c \in$ succ$|$CORRECT$_y(c|c_0)\}$
    - `// élément dont le produit scalaire avec` **w** `est maximal`
    - return B.top$_{\mathbf{w}}$, $gold$.top$_{\mathbf{w}}$
- gold $= \{c \in B|$CORRECT$_y(c|c_0)\}$
- return $B.top_{\mathbf{w}}$, $gold.top_{\mathbf{w}}$

depuis c_0^+ en ne suivant que les transitions de coût nul (ne coupant pas d'arc). On choisit alors comme configuration positive celle qui, parmi tous les antécédents de c_f^+, maximise l'écart de score avec c^-.

2.2 Reprise de décodage

Comme expliqué à la section 1.2, dans la stratégie EARLY UPDATE, le décodage d'un exemple est arrêté dès qu'une erreur est commise. Les configurations correspondant au début des dérivations sont donc sur-représentées dans l'ensemble d'apprentissage. Ainsi, sur le corpus français de SPMRL [5], seulement 41% des configurations finales sont atteintes à l'itération 10 de l'apprentissage et, en moyenne, les mises à jour ne prennent en compte que 57% des transitions d'une dérivation. La stratégie MAX VIOLATION corrige en partie ce biais en poursuivant l'exploration de l'espace de recherche après qu'une erreur a été commise. Toutefois, les mises à jour ne touchent que 53% des configurations finales et, en moyenne, ne prennent en compte que 81% d'une dérivation.

Pour éviter ce problème, nous proposons de modifier les stratégies EARLY UPATE et MAX VIOLATION pour les forcer à examiner les suffixes des dérivations qu'elles ignorent habituellement : après la mise à jour, le faisceau est réinitialisé à c^+ et le *décodage est poursuivi* jusqu'à ce qu'une nouvelle erreur soit détectée ou que la phrase soit complètement analysée ; de nouvelles mises à jour sont alors possibles. Cela revient, pour les deux stratégies, à modifier l'algorithme tel qu'indiqué par les algorithmes 2 et 3. La stratégie de recherche de la paire de configurations (c^-, c^+) reste inchangée.

2.3 Apprentissage depuis une configuration sous-optimale

Alors que l'apprentissage glouton statique ne visite que des configurations du chemin optimal, la recherche en faisceau permet d'en explorer un voisinage. Le classifieur aura donc déjà vu des

5. Nos conditions expérimentales sont détaillées dans la section 3.

Algorithme 2 : Apprentissage global standard.

```
Function DpTraining(dataset, T)
    w ← 0
    for i ∈ [[1, T]] do
        for x, y ∈ Shuffle(dataset)
        do
            c = Initial(x)
            c⁻, c⁺ = Oracle(c, y, w)
            w ← w + φ(c⁺) − φ(c⁻)
    return Average(w)
```

Algorithme 3 : Apprentissage global avec reprise.

```
c = Initial(x)
while ¬Final(c) do
    c⁻, c⁺ = Oracle(c, y, w)
    w ← w + φ(c⁺) − φ(c⁻)
    c ← c⁺
```

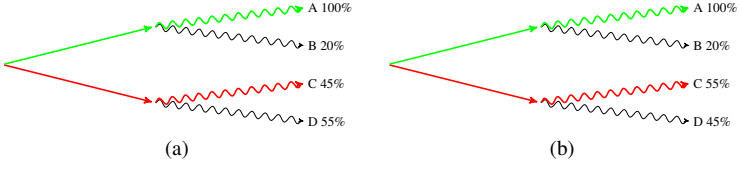

FIGURE 2: Situations indiscernables pour un classifieur entraîné sur des configurations optimales : c^- (rouge) est pénalisée, c^+ (vert) est renforcée ; mais la qualité relative de C et D est ignorée.

caractéristiques de configurations sous-optimales à l'apprentissage et l'entraînement à la récupération d'erreurs est, intuitivement, moins crucial dans ce cas. Pourtant, les chemins sous-optimaux n'étant jamais renforcés, le classifieur n'apprend pas à discriminer les dérivations sous-optimales selon leur qualité. La Figure 2 illustre sur une même phrase l'espace de recherche et la précision correspondant à chaque dérivation, dans deux situations opposées : dans la première, la dérivation D doit être favorisée par rapport à C, et inversement dans la deuxième. Or l'apprentissage se terminera dans les deux cas après la mise à jour de C vers A, sans avoir tiré parti de l'information de la qualité relative de C et D : C est pénalisée dans l'absolu donc aussi par rapport à D, et ce dans les deux cas.

Afin de renforcer la prise en compte des erreurs accumulées, nous proposons de modifier l'algorithme d'apprentissage en l'autorisant à débuter un décodage par une configuration arbitraire non optimale. Plusieurs stratégies peuvent être envisagées : tirer au hasard des configurations ou utiliser une configuration aléatoire au sein d'un décodage en cours. Dans le cas de la stratégie présentée dans le paragraphe précédent, il suffit de ré-initialiser le faisceau avec c^- au lieu de c^+. Cette modification nécessite uniquement un oracle complet, c.-à-d. pouvant être appelé depuis une configuration arbitraire, et c'est le cas de notre oracle (section 2.1), le calcul de Cost s'abstrayant des erreurs passées. Étant aussi non déterministe, il est en fait dynamique au sens de (Goldberg & Nivre, 2013).

Remarquons ici qu'en ayant adopté l'échelle de « temps » du décodage et en appliquant ces trois améliorations à un faisceau Early Update de taille 1, l'algorithme obtenu est exactement l'algorithme glouton dynamique de Goldberg & Nivre (2012).

3 Expériences

Cadre Nous étudions, dans cette section, les gains apportés par la combinaison des modifications proposées en § 2. Trois stratégies de référence sont considérées : l'analyseur glouton utilisant l'oracle dynamique de Goldberg & Nivre (2012) et les analyseurs en faisceau appris à l'aide des stratégies EARLY UPDATE et MAX VIOLATION.

Les analyseurs sont appris et testés sur les 9 langues de la campagne d'évaluation SPMRL (Seddah *et al.*, 2014). Dans toutes nos expériences, nous considérons un analyseur de type ARCEAGER avec une recherche en faisceau de taille 8 dont les paramètres sont estimés par un perceptron moyenné. Nous utilisons les parties du discours gros grain et les caractéristiques de (Zhang & Nivre, 2011) en ignorant les informations relatives aux étiquettes des dépendances mais en considérant les neuf dernières décisions. Ces caractéristiques ne prennent pas en compte l'aspect morphologiquement riche de ces langues. Chaque modèle est entraîné jusqu'à convergence sur un ensemble de développement.

Les performances sont évaluées par la précision des attachements non étiquetés (UAS) en ignorant la ponctuation. Pour évaluer la capacité des stratégies que nous proposons à mieux prédire les fins des dérivations, nous mesurons également l'UAS restreint à la prédiction des racines et l'UAS restreint à la prédiction des enfants et petits-enfants des racines, qui correspondent aux dernières dépendances attachées lors de la dérivation. Tous les scores sont moyennés sur 5 apprentissages.

Résultats La Table 1 rassemble les performances obtenues par chaque stratégie d'apprentissage. Ces résultats confirment que, comme mentionné dans de nombreux travaux, la recherche en faisceau améliore significativement la qualité de la prédiction par rapport au décodage glouton : elle permet de gagner $+0{,}5$ à $+2$ points suivant les langues et les stratégies d'apprentissage.

La combinaison de toutes les modifications proposées Section 2 permet d'obtenir des gains supplémentaires faibles mais consistants (entre $+0{,}1$ et $+0{,}5$) pour toutes les langues considérées. Il est intéressant de noter que, même c'est si presque toujours la combinaison de toutes les modifications proposées qui permet d'améliorer l'UAS, la cause de cette amélioration varie selon les langues considérées : pour l'arabe, elle est essentiellement due à la reprise après l'erreur (stratégie EARLY+ ND) alors que pour le coréen, c'est l'utilisation d'un oracle non déterministe (stratégie EARLY ND) qui a le plus fort impact. Comme attendu, les résultats de la Table 1 montrent que les modifications proposées permettent d'améliorer significativement la qualité des prédictions en fin de dérivation, surtout sur les petits corpus où l'exploitation de toute l'information disponible est cruciale : le gain de précision est souvent plus important pour les racines des arbres (choisies en toute fin de dérivation) et leurs enfants et petits-enfants que le gain moyen, jusqu'à $+1{,}5$ sur la racine en hébreu.

Pour MAX VIOLATION ces améliorations ont également un effet positif, mais plus faible que pour EARLY UPDATE, car le biais de sur-représentation est déjà réduit, possiblement aussi du fait de son oracle qui n'est pas purement non déterministe. Cet aspect reste à réexaminer dans des travaux futurs.

Analyse Les modifications des méthodes d'apprentissage que nous proposons visent principalement à accroître la similarité entre les configurations vues à l'apprentissage et en test. Pour vérifier cette intuition, nous proposons l'expérience suivante : si l'on suppose que les vecteurs de caractéristiques décrivant les configurations ne contiennent que des caractéristiques binaires engendrées indépendamment selon une distribution de Bernoulli, il est possible d'estimer la probabilité d'observation de chaque caractéristique sur l'ensemble d'apprentissage et sur l'ensemble de test et de calculer la

		de	ko	ar	fr	hg	ba	pl	he	sv
	#phrases	40 472	23 010	15 762	14 759	8 146	7 577	6 578	5 000	4 999
	écart-type	0,05	0,12	0,07	0,07	0,13	0,17	0,18	0,17	0,17
RES.	GREEDY DYN	90,73	82,79	83,98	84,23	84,33	84,00	87,66	83,78	86,35
	EARLY	92,74	82,73	85,03	86,02	85,63	84,42	89,60	85,39	87,00
		96,07 92,49	93,89 82,94	86,67 82,50	91,87 85,28	89,21 84,96	85,06 84,32	93,93 89,37	81,62 81,98	91,92 87,56
	MAXV	92,77	82,68	85,06	86,10	85,57	84,59	89,42	85,53	87,16
		96,16 92,54	93,97 82,86	86,89 82,11	91,97 85,09	88,86 84,98	85,43 84,53	93,95 89,12	82,38 81,89	91,57 87,51
AMÉLIORATIONS	EARLY ND	92,82	82,94	84,88	85,85	85,47	84,54	89,74	85,43	87,25
		96,11 92,60	93,98 83,13	86,50 82,53	91,43 85,11	88,65 84,70	85,12 84,39	94,13 89,41	81,04 81,94	92,14 87,91
	EARLY+ ND	92,80	82,80	85,21	86,18	85,62	84,52	89,80	85,40	87,15
		96,11 92,51	93,86 82,90	86,46 82,19	91,55 85,32	88,98 85,06	84,92 84,38	93,88 89,51	81,20 81,67	91,51 87,69
	EARLY+ ND SO	92,89	**82,98**	**85,27**	86,26	**85,74**	84,59	89,55	**85,84**	**87,37**
		96,11 92,62	94,02 83,04	86,46 82,59	91,68 85,48	88,84 85,12	85,16 84,39	94,29 89,29	83,11 82,35	92,90 87,92
	MAXV ND	92,82	82,71	85,09	86,13	85,42	84,47	89,67	85,50	87,24
		96,02 92,56	93,99 82,91	86,76 82,18	91,95 85,09	88,49 84,88	84,70 84,30	94,03 89,47	81,76 81,87	91,66 87,96
	MAXV+ ND	92,79	82,73	85,08	86,19	85,59	84,44	89,75	85,50	86,99
		96,04 92,51	94,00 82,95	86,95 82,07	91,89 85,17	88,51 84,94	85,02 84,28	94,03 89,48	81,96 81,89	91,70 87,51
	MAXV+ ND SO	**92,90**	82,94	85,04	**86,26**	85,55	**84,68**	**90,12**	85,83	87,31
		96,06 92,66	94,05 83,01	86,57 82,12	92,38 85,31	88,55 84,86	85,28 84,50	95,05 89,91	82,94 82,48	92,71 87,84

TABLE 1: UAS des différentes stratégies (PoS de référence) : GREEDY DYN (Goldberg & Nivre, 2013), early update (EARLY), max-violation (MAXV) et leurs variantes non-déterministes (ND), avec reprise (notées +) et en situation sous-optimale (SO). En deuxième ligne figurent l'UAS restreint aux racines et celui restreint à leurs enfants et petits-enfants.

	CLASSIQUE	ND	+ ND	+ ND SO
EARLY UPDATE	0,350	0,308	0,279	0,280
MAX VIOLATION	0,357	0,288	0,277	0,277

TABLE 2: Divergences de Kullback-Leibler entre les distributions de caractéristiques d'entraînement et de test, pour différentes stratégies d'apprentissage.

divergence de Kullback-Leibler [6] entre la distribution des vecteurs de caractéristiques en apprentissage et en test. La Table 2 montre cette similarité pour les différentes stratégies d'apprentissage dans le cas du français. Il apparaît que nos propositions permettent systématiquement de rendre la distribution des exemples en test plus proche de la distribution en apprentissage. Notons également que, de manière plus générale, les performances d'une stratégie d'apprentissage sont fortement corrélées avec la similarité entre distributions des exemples en test et en apprentissage.

6. En pratique, il est nécessaire de lisser les probabilités estimées afin de garantir que les distributions en test et en apprentissage ont même support. Dans nos expériences nous avons utilisé un lissage de Laplace de paramètre 0,1.

Efficacité La Figure 3 présente les courbes d'apprentissage en français de trois stratégies, suivant le nombre d'itérations, le nombre de mises à jour et le temps réel d'entraînement. Il apparaît que, malgré des mises à jour plus petites et donc plus nombreuses et individuellement moins efficaces, la stratégie de reprise égale et même surpasse MAX VIOLATION à oracle équivalent, en permettant la même accélération de convergence par rapport à EARLY UPDATE.

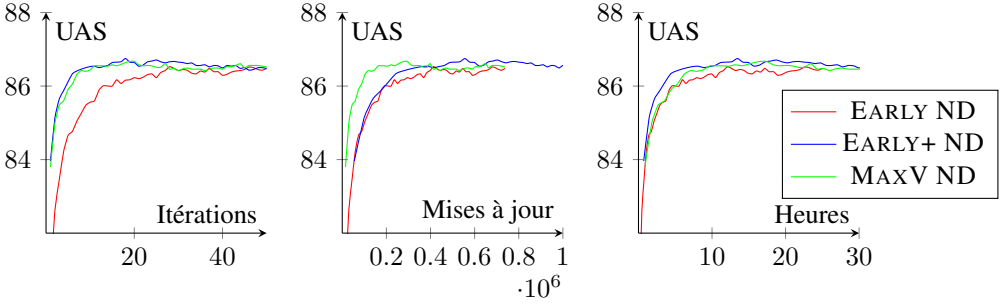

FIGURE 3: Courbes d'apprentissage en français, sur le corpus de développement.

4 Cadre commun pour l'étude des analyseurs par transition

Nous avons montré à la section 2 qu'en étendant le cadre de l'apprentissage global à plusieurs variantes, la recherche gloutonne devient comparable avec l'utilisation d'un faisceau de taille 1, faisant ainsi le pont entre deux stratégies habituellement considérées comme distinctes. Cette remarque nous motive pour poursuivre dans cette voie en cherchant un cadre général dans lequel s'inscrivent toutes les stratégies d'apprentissage de l'état de l'art. Dans un tel cadre, il devient possible de systématiser les comparaisons et ainsi d'identifier les déclinaisons de l'algorithme qui n'ont pas encore été étudiées.

4.1 Méta-algorithme pour l'analyse par transition

Les différentes stratégies s'unifient en les considérant à l'échelle d'une mise à jour, qui consiste en l'initialisation du faisceau, une phase de décodage puis, en cas d'erreur, le choix d'une paire de configurations sur lesquelles s'effectue la mise à jour. L'algorithme 4 décrit la procédure complète.

L'initialisation du faisceau à B_0, jusque là non traitée par la littérature, peut néanmoins varier en nombre et longueur des hypothèses comme en optimalité ; c'est en effet B_0 qui encode les erreurs passées accumulées, et donc la notion de correction d'une hypothèse finale. Son contenu initial conditionne le type d'information que l'on veut proposer au modèle.

Il s'ensuit un décodage normal dans le cadre d'une recherche en faisceau. Au mieux il sera écourté, en fonction d'un critère de détection anticipée d'erreur et d'optimisations de temps de calcul, sans que cela change la nature de la mise à jour attendue.

De nombreux paramètres entrent par contre en jeu pour la mise à jour proprement dite, dont l'objectif est d'extraire le maximum d'information du décodage (partiel) effectué. On identifie en réalité successivement, et en fonction de l'évolution du faisceau, du modèle et du moment de détection de l'erreur, une série de configurations positives (resp. négatives) de profondeur croissante, puis

on choisit la paire de configurations qui sera visée par la mise à jour. Outre les variations déjà citées, la génération des configurations candidates est sujette à une alternative : soit on génère une seule configuration finale, dont on considère les antécédents, soit on choisit des configurations indépendantes pour chaque profondeur, renforçant ainsi l'importance des hypothèses courantes.

Enfin, la mise à jour globale étant décomposable suivant chaque paire de transitions, celles-ci peuvent être pondérées différemment. Bien que cela soit sans effet en perceptron standard sur des données linéairement séparables, et donc non utilisé dans l'état de l'art de l'analyse en dépendances ($\lambda_i = 1$), il reste à étudier, dans le cas non séparable, les conséquences sur les performances d'une mise à jour adaptée et ses interactions avec le perceptron moyenné.

Le cadre présenté ici permet, en replaçant les mises à jour au cœur de l'algorithme, de se défaire des présuppositions sur leur nombre ou leur position par rapport à la phrase. Par ailleurs, imposer une spécification complète de la stratégie sans simplification *ad hoc* permet de comparer deux stratégies de complexités différentes. On remarque ainsi que malgré une description bien plus simple, la stratégie EARLY UPDATE ne diffère de MAX VIOLATION que par sa valeur n_{update} fixée à n_0 au lieu de $n_{\text{écart max}}$: dans les deux cas on initialise B_0 à une seule configuration vide, n_0 marque la sortie de la référence du faisceau, et les spécifications de c^+ et c^- données plus haut pour MAX VIOLATION conviennent aux deux stratégies. La stratégie de reprise correspond quant à elle simplement à une initialisation de B_0 avec une configuration non vide choisie de manière idoine.

Algorithme 4 : Algorithme général de mise à jour.

Function UPDATE(c_0, *y*, $\mathbf{w}$)
  $B_0 \leftarrow \{c_1, c_2, ..., c_{k'}\}, k' \leq k$ au temps t_0
  $Correct = \{$meilleures configurations finales accessibles depuis $B_0\}$
  $B_1, B_2, ..., B_N \leftarrow$ DÉCODAGE(B_0, w^{t_0}, k) tel que FINAL(B_N)
  if $B_N.top \notin Correct$ **then**
    On le savait depuis B_{n_0}. On choisit successivement :
    • une configuration c^+ pour chaque profondeur, sachant $\{B_i\}_i, n_0, w^{t_0}, Correct$
    • une configuration c^- pour chaque profondeur, sachant $\{B_i\}_i, n_0, w^{t_0}, \{c^+\}$
    • une profondeur n_{update}, sachant $\{B_i\}_i, n_0, w^{t_0}, \{c^+\}, \{c^-\}$
    $\mathbf{w^{t_0+1}} \leftarrow \mathbf{w^{t_0}} + \sum_{i=0}^{n_{update}} \lambda_i(\phi(transition_i(c^+_{n_{update}})) - \phi(transition_i(c^-_{n_{update}})))$

4.2 Autres limites

L'esquisse de ce schéma général d'apprentissage permet encore d'identifier d'importantes limites des stratégies classiques, qui ouvrent de multiples perspectives de recherche :

— Les interactions entre recherche en faisceau et apprentissage, déjà visibles dans les travaux de Zhang & Nivre (2012), avec d'importantes chutes de performances lorsque l'on décorrèle les deux, confirment que le faisceau n'est pas une simple juxtaposition d'hypothèses, mais un objet complexe prédit dans son ensemble par le classifieur. Les mises à jour sont pourtant faites sur une seule paire de configurations, négligeant de fait les interactions entre hypothèses.

— La détection des erreurs dans l'état de l'art utilise un critère simple, mais qui ne garantit pas pour autant que l'erreur vient juste d'être commise, comme l'illustre la Figure 4. Il manque en réalité, suite à cette détection, une étape d'analyse du comportement récent du faisceau pour estimer quand l'erreur a vraiment été commise.

— À travers la description et l'implémentation des critères d'apprentissage transparaissent deux fonctions distinctes du modèle : d'une part conserver les références au sein du faisceau de recherche, d'autre part classer des configurations finales en favorisant le meilleur UAS. Cette double fonction montre l'importance d'étudier des algorithmes combinant deux modèles, suivant ces deux objectifs : d'après une expérience oracle de réordonnancement *a posteriori* des hypothèses, nos modèles ont tous une marge d'amélioration de 4 points.

— Enfin, le décodage est effectué par extensions successives du faisceau entier, donc de manière synchrone par rapport aux longueurs des hypothèses, ce qui mène à comparer des configurations sans caractéristiques communes car couvrant des portions de phrases différentes. L'approche EARLY UPDATE ayant été conçue par Collins & Roark (2004), qui utilise un décodage synchrone par rapport à l'avancée du *buffer*, il est envisageable d'obtenir des gains supplémentaires avec une synchronisation différente des hypothèses.

Toutes ces considérations soulignent le faible nombre de pistes de recherche déjà exploitées dans la littérature sur ce sujet, eu égard à la grande variabilité des principales stratégies de l'état de l'art. Quoique déjà performants, les apprentissages en faisceau ne semblent donc pas parvenus au maximum de leurs performances possibles, et les opportunités de recherche sont encore nombreuses.

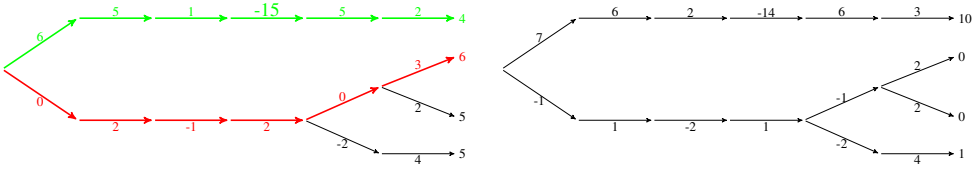

FIGURE 4: Mauvaise estimation de la profondeur d'erreur : espace de recherche avant et après mise à jour de la dérivation rouge vers la verte (faisceau de taille 3). Le score global figure en fin de dérivation et le score local sur chaque transition. La chute brutale du score de la référence n'est pas détectée et le score relatif des trois autres dérivations est modifié, alors qu'elles ne divergent qu'après.

5 Conclusion

À travers trois modifications (oracle non déterministe, reprise de décodage et apprentissage en sous-optimal), nous avons montré qu'il était possible d'améliorer les performances des stratégies classiques pour l'apprentissage d'analyseurs en dépendances par transition, sans pour autant changer ni l'esprit de ces stratégies, ni leur complexité. Ces algorithmes intègrent de nombreux choix de conception, en apparence mineurs et négligés par la littérature, mais dont une combinaison adéquate peut faire gagner jusqu'à 0,5 en UAS. Une mesure simple de similarité a permis d'expliquer ces performances comme un effet du rapprochement des configurations de test et d'apprentissage. Le formalisme introduit ouvre plusieurs perspectives pour ces travaux, que facilitera une description unifiée.

Remerciements

Ces travaux ont été partiellement financés par la Direction générale de l'armement. Nous remercions les relecteurs pour leurs commentaires pertinents et détaillés sur l'article.

Références

AUFRANT L. & WISNIEWSKI G. (2016). *PanParser : a Modular Implementation for Efficient Transition-Based Dependency Parsing*. Technical report, LIMSI-CNRS.

CHANG K., HE H., DAUMÉ III H. & LANGFORD J. (2015). Learning to Search for Dependencies. *CoRR*, **abs/1503.05615**.

COAVOUX M. & CRABBÉ B. (2015). Comparaison d'architectures neuronales pour l'analyse syntaxique en constituants. In *Actes de la 22e conférence sur le Traitement Automatique des Langues Naturelles*, p. 293–304, Caen, France : Association pour le Traitement Automatique des Langues.

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

COLLINS M. & ROARK B. (2004). Incremental Parsing with the Perceptron Algorithm. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, p. 111–118, Barcelona, Spain.

GOLDBERG Y. & NIVRE J. (2012). A Dynamic Oracle for Arc-Eager Dependency Parsing. In *Proceedings of COLING 2012*, p. 959–976.

GOLDBERG Y. & NIVRE J. (2013). Training Deterministic Parsers with Non-Deterministic Oracles. *Transactions of the Association for Computational Linguistics*, p. 403–414.

HOPKINS M. & MAY J. (2011). Tuning as Ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, p. 1352–1362, Edinburgh, Scotland, UK. : Association for Computational Linguistics.

HUANG L., FAYONG S. & GUO Y. (2012). Structured Perceptron with Inexact Search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, p. 142–151.

KNYAZEVA E., WISNIEWSKI G. & YVON F. (2015). Apprentissage par imitation pour l'étiquetage de séquences : vers une formalisation des méthodes d'étiquetage « easy-first ». In *Actes de la 22e conférence sur le Traitement Automatique des Langues Naturelles*, p. 1–12, Caen, France.

NIVRE J. (2008). Algorithms for Deterministic Incremental Dependency Parsing. *Comput. Linguist.*, **34**(4), 513–553.

OCH F. J. & NEY H. (2002). Discriminative Training and Maximum Entropy Models for Statistical Machine Translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, p. 295–302, Philadelphia, Pennsylvania, USA.

ROSS S. & BAGNELL D. (2010). Efficient Reductions for Imitation Learning. In *Proceedings of the International Conference on Artificial Intelligence on Statistics*, AISTATS'10, p. 661–668.

SEDDAH D., KÜBLER S. & TSARFATY R. (2014). Introducing the SPMRL 2014 Shared Task on Parsing Morphologically-rich Languages. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, p. 103–109, Dublin, Ireland : Dublin City University.

XU Y., FERN A. & YOON S. (2009). Learning Linear Ranking Functions for Beam Search with Application to Planning. *J. Mach. Learn. Res.*, **10**, 1571–1610.

ZHANG Y. & CLARK S. (2008). A Tale of Two Parsers : Investigating and Combining Graph-based and Transition-based Dependency Parsing Using Beam-search. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, p. 562–571, Stroudsburg, PA, USA : Association for Computational Linguistics.

ZHANG Y. & NIVRE J. (2011). Transition-based Dependency Parsing with Rich Non-local Features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics : Human Language Technologies*, p. 188–193.

ZHANG Y. & NIVRE J. (2012). Analyzing the Effect of Global Learning and Beam-Search on Transition-Based Dependency Parsing. In *Proceedings of COLING 2012 : Posters*, p. 1391–1400.