Apprentissage par imitation pour l'étiquetage de séquences : vers une formalisation des méthodes d'étiquetage « *easy-first* »

Elena Knyazeva[1, 2] Guillaume Wisniewski[1, 2] François Yvon[2]
(1) Université Paris Sud, 91 403 Orsay CEDEX
(2) LIMSI-CNRS, 91 403 Orsay CEDEX
{nom.prénom}@limsi.fr

Résumé. De nombreuses méthodes ont été proposées pour accélérer la prédiction d'objets structurés (tels que les arbres ou les séquences), ou pour permettre la prise en compte de dépendances plus riches afin d'améliorer les performances de la prédiction. Ces méthodes reposent généralement sur des techniques d'inférence approchée et ne bénéficient d'aucune garantie théorique aussi bien du point de vue de la qualité de la solution trouvée que du point de vue de leur critère d'apprentissage.
Dans ce travail, nous étudions une nouvelle formulation de l'apprentissage structuré qui consiste à voir celui-ci comme un processus incrémental au cours duquel la sortie est construite de façon progressive. Ce cadre permet de formaliser plusieurs approches de prédiction structurée existantes. Grâce au lien que nous faisons entre apprentissage structuré et apprentissage par renforcement, nous sommes en mesure de proposer une méthode théoriquement bien justifiée pour apprendre des méthodes d'inférence approchée. Les expériences que nous réalisons sur quatre tâches de TAL valident l'approche proposée.

Abstract.
Imitation learning for sequence labeling: towards a formalization of easy-first labeling methods.
Structured learning techniques, aimed at modeling structured objects such as labeled trees or strings, are computationally expensive. Many attempts have been made to reduce their complexity, either to speed up learning and inference, or to take richer dependencies into account. These attempts typically rely on approximate inference techniques and usually provide very little theoretical guarantee regarding the optimality of the solutions they find.
In this work we study a new formulation of structured learning where inference is primarily viewed as an incremental process along which a solution is progressively computed. This framework generalizes several structured learning approaches. Building on the connections between this framework and reinforcement learning, we propose a theoretically sound method to learn to perform approximate inference. Experiments on four sequence labeling tasks show that our approach is very competitive when compared to several strong baselines.

Mots-clés : Apprentissage par Imitation ; Apprentissage Structuré ; Étiquetage de Séquences.

Keywords: Imitation Learning ; Structured Learning ; Sequence Models.

1 Introduction

L'apprentissage structuré a pour objectif de prédire des objets composés de parties [1] inter-dépendantes, comme c'est le cas pour des arbres ou des séquences, en exploitant les dépendances entre parties pour améliorer les performances des prédictions. La prédiction d'objets structurés est au cœur de nombreuses tâches de Traitement Automatique des Langues (TAL) : analyse syntaxique en dépendances, reconnaissance d'entités nommées, traduction automatique, etc. La quasi totalité des modèles d'apprentissage structuré, tels que les grammaires hors-contexte probabilistes (PCFG) ou les champs aléatoires conditionnels (CRF), reposent sur une généralisation de la classification multi-classes : ils visent en effet à apprendre une fonction de score (par exemple une probabilité jointe ou conditionnelle) mesurant l'adéquation entre une observation et chacune des structures possibles. Une fois apprise, cette fonction permet, lors de la prédiction, de distinguer la meilleure solution parmi toutes les hypothèses considérées. Notre travail propose une formulation alternative

1. Dans la suite, nous considérons que chaque partie de la structure à prédire peut être représenté par une étiquette choisie dans un inventaire fini.

de l'apprentissage structuré, issue du modèle de Collins & Roark (2004). Plutôt que de modéliser les propriétés d'une « bonne » solution, puis de chercher cette solution parmi un ensemble de candidats, cette formulation consiste à modéliser directement le processus permettant de *construire* une solution correcte. Elle permet de contourner plusieurs limites des modèles d'apprentissage structuré classiques (Daumé III & Marcu, 2005).

En effet, les modèles d'apprentissage structuré classiques souffrent de deux problèmes majeurs. Premièrement, leur expressivité est limitée : pour pouvoir résoudre le problème (combinatoire) de la recherche de la meilleure sortie, il est en effet nécessaire de choisir une paramétrisation idoine de la fonction de score, qui ne considère que des dépendances *locales* entre étiquettes (hypothèse de Markov). La programmation dynamique permet alors de trouver efficacement, par exemple à l'aide de l'algorithme de Viterbi, la solution de meilleur score. En conséquence, la plupart des méthodes d'apprentissage structuré de l'état de l'art ne peuvent pas tirer pleinement profit de toutes les informations contenues de structure. Deuxièmement, même lorsque seules des dépendances locales sont considérées, la complexité des méthodes d'apprentissage structuré reste souvent élevée, surtout pour des tâches présentant un grand nombre d'étiquettes.

Les méthodes de décodage approché reposant, par exemple, sur des stratégies de recherche gloutonne ou sur des algorithmes comme A*, constituent une manière naturelle d'éviter ces deux problèmes : elles permettent soit de considérer des paramétrisations du score pour lesquelles il n'existe pas de méthode d'inférence efficace, soit, lorsque seules des dépendances locales sont considérées, d'accélérer le décodage. Les performances en prédiction de ces approches sont toutefois limitées par le phénomène de propagation des erreurs : comme le choix des étiquettes est, en partie, fondé sur le choix des étiquettes précédentes (qui n'est généralement pas remis en cause), une erreur de prédiction complique les choix futurs et la probabilité de commettre une erreur augmente donc rapidement au fur et à mesure que les décisions sont prises.

Deux familles de méthodes ont été proposées dans la littérature pour limiter la propagation d'erreurs lors d'un décodage avec une méthode heuristique. Les méthodes de la première famille cherchent à modifier la distribution des exemples d'apprentissage en fonction des erreurs qui sont faites lors du décodage, afin d'apprendre à faire des prédictions correctes même en présence d'erreurs dans les décisions passées. Ainsi, Finkel *et al.* (2006) proposent une stratégie fondée sur la modélisation de trajectoires engendrées par le décodeur en utilisant une méthode de Monte Carlo. Cette approche bénéficie de garanties théoriques fortes, mais sa complexité rend sa mise en œuvre prohibitive. Daumé III *et al.* (2009) décrivent un modèle similaire dans un cadre d'apprentissage par renforcement. Notre travail s'inscrit dans la continuité de cette étude.

Les méthodes de la seconde famille, généralement qualifiées de « plus simple d'abord » *(easy first)*, modifient le décodage afin de retarder les décisions les moins sûres, limitant ainsi le risque de propagation d'erreurs. Par exemple, Shen *et al.* (2007) décrivent un analyseur morpho-syntaxique capable de prédire les étiquettes dans un ordre libre ; Yamada & Matsumoto (2003) proposent un analyseur en dépendances qui s'appuie sur une méthode d'inférence gloutonne et Goldberg & Elhadad (2010) généralisent cette approche pour construire les dépendances en ordre libre. Finalement, Gesmundo & Henderson (2014) appliquent ce principe à la traduction automatique hiérarchique (Chiang, 2007) en relâchant la contrainte de réaliser l'inférence de manière ascendante. Bien que ces méthodes aient obtenu des résultats expérimentaux concluants, elles ne bénéficient d'aucune garantie théorique et leur mise en œuvre repose sur plusieurs heuristiques.

En prenant pour exemple la tâche d'étiquetage de séquences, nous proposons, dans ce travail, un nouveau cadre qui permet de réunir ces deux familles dans un formalisme commun (§ 2). En faisant le lien entre ce cadre et l'apprentissage par renforcement, nous proposons une méthode théoriquement bien justifiée pour apprendre des méthodes de décodage *easy first* (§ 3). Les expériences réalisées sur quatre tâches de TAL (§ 4) montrent que la méthode proposée permet d'améliorer légèrement les résultats de l'état-de-l'art, tout en réduisant la complexité de l'inférence et de l'apprentissage.

2 Apprentissage structuré incrémental

Dans cette section, nous introduisons un cadre général pour l'apprentissage structuré incrémental (§ 2.1) qui permet de reformuler le problème de l'apprentissage structuré ; nous présentons ensuite deux modèles d'étiquetage de séquences (§ 2.2) qui instancient ce cadre.

2.1 Principes

Le principe fondamental des algorithmes d'apprentissage structuré *incrémental* est de considérer un espace de recherche regroupant l'ensemble des structures possibles, que l'ensemble des sous-parties de celles-ci. Par exemple, dans le cas de l'étiquetage de séquences, l'espace de recherche considéré regroupera tous les étiquetages possibles de la séquence d'ob-

servations complète, ainsi que tous les étiquetages *partiels* de celle-ci (c.-à-d. dans lesquels seules certaines observations sont étiquetées).

L'espace de recherche, muni de la relation d'ordre suivante : xRy si x est une sous-partie de y, définit alors un semi-treillis. Cette définition permet de reformuler la recherche de la structure de plus grand score comme un processus de construction : dans le treillis décrivant l'espace de recherche, chaque nœud correspond à une solution partielle et une arête dénote un moyen d'*étendre* cette solution vers une solution plus complète. Il est ainsi possible de considérer l'ensemble des arrêtes issues d'un nœud comme un ensemble d'actions réalisables à partir de ce nœud : le parcours de l'espace de recherche peut alors être décrit comme un processus de construction qui va incrémentalement étendre une solution partielle vers une solution complète. L'inférence se résume alors à une suite d'*actions* ou de *décisions* (quelle solution partielle étendre ? comment l'étendre ? etc.)

Plus formellement, un espace de recherche peut être défini, en intension, par un Processus de Décision Markovien (PDM) (Sutton & Barto, 1998). Dans ce travail [2], un PDM correspond à un quintuplet $\langle \mathcal{S}, \mathcal{A}, R, \mathcal{S}_f, s_i \rangle$:

- $\mathcal{S}$ est l'ensemble des états possibles (l'ensemble des séquences d'étiquettes complètes ou non) ;
- $\mathcal{A}_s$ est l'ensemble des actions réalisables dans l'état s ; chacune de ces actions permet de passer dans un état s' en « étendant » la structure partielle décrite par l'état s ; on notera $s' = s \oplus a$ pour noter l'opération d'extension ;
- r est une fonction de $\mathcal{S} \times \mathcal{A}$ dans $\mathbb{R}$ qui définit la *récompense* $r(a,s)$ reçue lorsque l'action a est exécutée dans l'état s ;
- $\mathcal{S}_f \subset \mathcal{S}$ est l'ensemble des états finaux ;
- $s_i \in \mathcal{S}$ est un état initial (supposé unique).

La fonction de récompense peut être choisie de manière arbitraire. Il est cependant naturel de la lier à la qualité (telle qu'évaluée par la fonction de coût du problème) de la solution partielle que permet d'obtenir l'action a. Dans ce cas toutefois, les récompenses ne peuvent être connues que pour des données étiquetées, puisqu'elles s'appuient sur l'évaluation (partielle) de la fonction de coût. Déterminer un apprenant capable de prédire ces récompenses pour les données de test est au cœur de l'approche que nous proposons ici (§3).

Une *politique* $\pi : \mathcal{S} \mapsto \mathcal{A}$ est une fonction qui détermine l'action $a_t = \pi(s_t)$ qui doit être choisie dans un état s_t donné. Elle spécifie une suite d'actions $a_1, ..., a_T$, qui permet d'atteindre un état final à partir de l'état initial en construisant une séquence d'étiquettes $\mathbf{y}$. Une politique, et par conséquent, la séquence d'étiquettes qu'elle engendre, est associée à une *récompense cumulée* définie par :

$$V(\pi) = \sum_{t \in [\![0,T]\!]} r(s_t, \pi(s_t)) \tag{1}$$

où s_0 est l'état initial et s_T est un état final.

Étant donné un PDM, l'apprentissage par renforcement cherche à déterminer la *politique optimale*, c'est-à-dire celle dont la récompense cumulée est la plus grande. Il existe, pour les « petits » PDM, des méthodes, fondées sur la programmation dynamique, comme les algorithmes *Value Iteration* et *Policy Iteration* (Sutton & Barto, 1998) qui permettent de construire la politique optimale efficacement. Mais, dans le cas général, quand l'espace de recherche ne peut plus être exploré entièrement, il faut s'en remettre à des méthodes de recherche approchée. Nous adoptons, dans ce travail, une méthode qui repose sur la politique gloutonne :

$$a_t = \pi(s_t) = \underset{a \in \mathcal{A}_{s_t}}{\arg\max}\, f_\pi(s_t, a) \tag{2}$$

où $f_\pi(s_t, a)$ est une fonction de score associée à la politique π, dépendant uniquement de l'action a et de l'état courant s_t. La fonction de score doit être choisie de sorte que la réalisation de la politique associée obtienne une récompense cumulée $V(\pi)$ élevée. Nous verrons, à la section 3 comment l'apprentissage par imitation permet d'estimer conjointement la fonction de score et la fonction de récompense.

Avec une stratégie gloutonne, la réalisation d'une politique (qui permet d'engendrer la séquence d'étiquettes) possède une complexité linéaire par rapport au nombre d'états et d'actions, tant que la fonction de score n'utilise que des informations contenues dans l'état courant. Elle n'offre toutefois aucune garantie concernant l'optimalité de la solution trouvée.

2. Dans la définition usuelle des PDM, lors de la réalisation d'une action a dans un état s le système passe dans un état s' avec une certaine probabilité. Cette généralisation permet de modéliser une interaction plus réaliste avec l'environnement. Par exemple, en robotique, lorsque le système décide de réaliser l'action « tourner à gauche de 30° » il est possible qu'à causes des frottements ou d'obstacles il ne se retrouve pas exactement dans l'état envisagé. Par souci de clarté, nous simplifions nos définitions au cas déterministe qui est suffisant pour décrire notre travail.

2.2 Modèles d'étiquetage de séquences

Nous allons introduire, dans cette section, deux PDM permettant de réaliser un étiquetage de séquences. Dans la suite, nous noterons $\mathbf{x} = (x_t)_{t=1}^T$ une séquence de T observations, $\mathbf{y} = (y_t)_{t=1}^T$ la séquence d'étiquettes correspondante avec $x_t \in \mathcal{X}$ et $y_t \in \mathcal{Y}$. Nous utiliserons également les notations $\mathbf{x}_\theta$, où θ est un ensemble d'entiers compris entre 1 et T, pour désigner les éléments de $\mathbf{x}$ dont les indices sont dans θ ; et $i{:}j$ pour l'ensemble des entiers compris entre i et j (inclus).

2.2.1 Étiquetage (monotone) gauche-droite

Le premier modèle d'étiquetage considéré correspond au PDM suivant :

— $\mathcal{S} = \left\{(\mathbf{x}_{1:T}, \mathbf{y}_{1:t}), \mathbf{x}_{1:T} \in \mathcal{X}^T, \mathbf{y}_{1:t} \in \mathcal{Y}^t, 0 \leq t \leq T\right\}$: chaque état est décrit par l'observation et une sortie partielle dont seules les t premières étiquettes sont prédites ;

— $\mathcal{A}_s = \{y, y \in Y\}$: chaque action ajoute une nouvelle étiquette à une séquence dont les t premières étiquettes sont connues et permet de passer de l'état $s = (\mathbf{x}_{1:T}, \mathbf{y}_{1:t})$ à l'état $s' = (\mathbf{x}_{1:T}, \mathbf{y}_{1:t+1})$ avec $\mathbf{y}_{1:t+1} = \mathbf{y}_{1:t} \oplus y$;

— $\mathcal{S}_f = \left\{(\mathbf{x}_{1:T}, \mathbf{y}_{1:T}), \mathbf{x}_{1:T} \in \mathcal{X}^T, \mathbf{y}_{1:T} \in \mathcal{Y}^T\right\}$: les états finaux correspondent à l'ensemble des séquences complètement étiquetées ;

— $s_i = (\mathbf{x}_{1:T}, \emptyset)$: l'état initial correspond à une séquence d'étiquettes vide.

La fonction de récompense sera définie au § 3.1. Ce PDM comporte $\sum_{t=0}^T |\mathcal{Y}|^t$ états et $\sum_{t=0}^T |\mathcal{Y}|^t \times |\mathcal{Y}|$ actions ($|\mathcal{Y}|$ actions par état). Il permet de construire une solution en choisissant, successivement de gauche à droite, l'étiquette de chaque observation et décrit un espace de recherche qui correspond à celui considéré par l'algorithme de Viterbi (et plus généralement par les autres algorithmes utilisant la programmation dynamique comme l'algorithme *forward-backward*). La Figure 1 en donne un exemple.

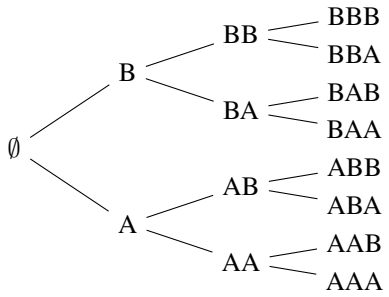

FIGURE 1 – Espace de recherche correspondant à l'étiquetage gauche-droite d'une séquence de trois observations lorsqu'il y a deux étiquettes possibles (A et B). Pour simplifier, la représentation des états n'inclut pas la séquence d'observations.

Ce modèle et le cadre d'apprentissage associé offrent deux avantages par rapport aux modèles de séquences classiques comme les HMM ou les CRF. Premièrement la complexité de l'inférence (en $\mathcal{O}\left(T \cdot |\mathcal{Y}|\right)$) est plus faible que celle des modèles qui se basent sur un décodage de Viterbi (en $\mathcal{O}\left(T \cdot |\mathcal{Y}|^2\right)$). Deuxièmement, il est possible d'introduire, sans changer la complexité de l'inférence ou de l'apprentissage, des caractéristiques décrivant toutes les décisions passées (ces informations sont directement déductibles des informations stockées dans chaque état) alors que, dans un CRF linéaire d'ordre 1, seules des caractéristiques décrivant l'étiquette précédente peuvent être incluses [3]. L'inférence sera toutefois approximative : rien ne garantit que la solution trouvée correspond effectivement à celle qui a le plus grand score.

2.2.2 Étiquetage en ordre libre

Le second modèle d'étiquetage que nous considérons est une généralisation du modèle précédent, *qui n'impose plus l'ordre dans lequel les positions étiquetées sont choisies*. Il correspond au PDM suivant :

— $\mathcal{S} = \{(\mathbf{x}_{1:T}, \mathbf{y}_\theta), \theta \in \wp([\![1:T]\!])\}$ où $\wp(E)$ est l'ensemble des parties de E : chaque état correspond à une séquence dont seuls les éléments dont les indices sont dans θ sont étiquetés ; les positions étiquetées ne sont pas nécessairement contiguës ;

3. Inclure des caractéristiques d'ordre supérieur dans des CRF augmente la complexité de l'apprentissage et de l'inférence exponentiellement en l'ordre du modèle.

— $\mathcal{A}_s = \{(y,t), y \in Y, t \notin \theta\}$ pour $s = (\mathbf{x}_{1:T}, \mathbf{y}_\theta)$: une action choisit *une des positions non étiquetées et son étiquette* et permet de passer de l'état $(\mathbf{x}_{1:T}, \mathbf{y}_\theta)$ à l'état $(\mathbf{x}_{1:T}, \mathbf{y}_{\theta'})$ avec $\theta' = \theta \cup \{t\}$ et $y_t = y$;
— $\mathcal{S}_f = \left\{(\mathbf{x}_{1:T}, \mathbf{y}_{1:T}), \mathbf{x}_{1:T} \in \mathcal{X}^T, \mathbf{y}_{1:T} \in \mathcal{Y}^T\right\}$: les états finaux correspondent à l'ensemble des séquences totalement étiquetées ;
— $s_0 = (\mathbf{x}_{1:T}, \emptyset)$: l'état initial correspond à une séquence d'étiquettes vide.

La fonction de récompense est définie en § 3.1. Ce modèle permet d'étiqueter une séquence en choisissant l'ordre dans lequel les étiquettes sont attribuées. Il repose sur l'intuition que certaines étiquettes sont plus faciles à prédire que d'autres et que la connaissance de celles-ci facilitera le choix des autres étiquettes. Comme pour le modèle gauche-droite, les informations stockées dans chaque état permettent de définir des caractéristiques plus riches (dépendances longues, étiquettes « futures », ...) que celles prises en compte dans les modèles standard de prédiction de séquences.

La figure 2 donne un exemple de l'espace de recherche considéré lors d'un étiquetage en ordre libre. De manière générale, celui-ci comporte $\sum_{t=0}^{T} \binom{t}{T} |\mathcal{Y}|^t$ états et $\sum_{t=0}^{T} \binom{t}{T} |\mathcal{Y}|^t \times |\mathcal{Y}|$ actions. On notera que lorsque la contrainte sur l'ordre des étiquettes est relâchée, plusieurs séquences d'actions peuvent conduire dans le même état.

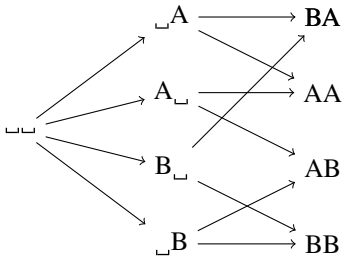

FIGURE 2 – Espace de recherche d'un étiquetage en ordre libre d'une séquence de deux observations lorsqu'il y a deux étiquettes possibles. Les positions dont l'étiquette n'est pas encore prédite sont indiquées par le symbole ␣.

3 Apprentissage de la fonction de score

3.1 Apprentissage par imitation

Dans le cadre de l'apprentissage structuré incrémental, l'objectif de l'apprentissage est d'estimer la fonction de récompense à partir d'un ensemble de séquences étiquetées, puis la fonction de score f sur laquelle la politique gloutonne est fondée. La fonction de score devra être choisie de sorte que *i)* la récompense cumulée de la politique soit d'autant plus élevée que la solution est de bonne qualité (telle qu'évaluée par une fonction de coût dépendant de la tâche, comme le score F_1 ou la distance de Hamming) *ii)* en intégrant le fait que les décisions sont prises de manière gloutonne et, donc que le score doit inclure une estimation des coûts futurs pour que la solution trouvée soit proche de la solution optimale.

Plusieurs méthodes répondant à ces deux objectifs ont été proposées en apprentissage par renforcement. Dans ce travail nous nous intéressons aux méthodes dites d'*apprentissage par imitation (imitation learning)* (Abbeel & Ng, 2004; Ross & Bagnell, 2010). Ces méthodes estiment la fonction de score et de récompense conjointement en *réduisant* ces deux problèmes à un problème de classification multi-classes (Langford & Zadrozny, 2005) : le parcours de l'espace de recherche est vu comme une suite de problèmes de classification dont l'objectif est de retrouver, dans chaque état, la meilleure action possible, correspondant à celle que prendrait un oracle. Il est raisonnable de supposer que cette action est celle dont le choix permettra d'obtenir, *dans l'état final*, une solution dont la récompense *cumulée* sera optimale. L'exemple canonique motivant ce cadre est l'apprentissage d'un système conduisant une voiture à partir de l'observation du comportement d'un conducteur (Ross & Bagnell, 2010).

Plus précisement, nous considérons, pour choisir l'action à effectuer dans un état s, un classifieur linéaire multi-classes. La fonction de score de la règle de décision décrite dans l'équation (2) s'écrit alors :

$$f(s,a) = \langle \phi(a,s) \,|\mathbf{w}\rangle \tag{3}$$

où $\langle\cdot|\cdot\rangle$ est le produit scalaire usuel, $\mathbf{w}$ est le vecteur de paramètres du classifieur, $\phi(a,s)$ est un vecteur de caractéristiques décrivant l'action a et l'état courant s. Comme l'état encode toutes les informations concernant la séquence d'observations

et la solution partiellement étiquetée, il est possible de définir des caractéristiques riches prenant, par exemple, en compte des historiques de grande taille ou, dans le cas du modèle en ordre libre, les étiquettes des voisins déjà prédites.

L'apprentissage de ce classifieur repose sur la connaissance de la *politique oracle* qui associe, à chaque état, la meilleure action pouvant être choisie. Il suffit alors d'utiliser le corpus d'apprentissage pour engendrer l'ensemble des états et des actions correspondants, d'étiqueter ceux-ci grâce à la politique oracle (c.-à-d. de déterminer, pour chaque exemple, quelle est l'action qui aurait dû être choisie) puis d'estimer les paramètres du classifieur à l'aide d'un algorithme d'apprentissage supervisé standard. La mise en œuvre de ce principe soulève toutefois plusieurs problèmes :

1. Étant donnée la taille de l'espace de recherche, il est impossible de considérer tous les états possibles comme exemples lors de l'apprentissage ;
2. Dans le cadre proposé, le modèle de séquences est appris en optimisant une fonction de coût 0/1 qui permet de caractériser la capacité de l'apprenant à choisir la bonne action dans chaque état. Cette fonction objectif n'a aucun lien direct avec la qualité des séquences prédites et les garanties offertes par la théorie de l'apprentissage statistique [4] ne s'appliquent donc pas. L'optimisation d'un coût 0/1 en apprentissage est-elle théoriquement fondée ?
3. Il n'est pas toujours possible, lors de l'apprentissage, d'avoir accès directement à un oracle indiquant quelle action choisir dans un état donné ; celui-ci devra être reconstruit à partir de la référence.

La plupart des méthodes *easy first* de la littérature ignorent ces question ou n'y apportent qu'une réponse expérimentale. Le lien que nous faisons ici avec l'apprentissage par imitation, et plus généralement l'apprentissage par renforcement, permet d'obtenir des garanties théoriques. En effet, les deux premiers problèmes sont des problèmes intrinsèques à l'apprentissage par imitation et plusieurs solutions (Langford & Zadrozny, 2005; Daumé III *et al.*, 2009; Ross & Bagnell, 2010) ont été proposées dans la littérature. Nous détaillons une de ces solutions, SEARN, dans la section suivante. Le dernier problème est un problème spécifique à l'application de l'apprentissage par imitation à la prédiction de séquences auquel nous proposerons une nouvelle solution présentée dans la section 3.3.

3.2 SEARN

Nous résumons, dans cette section, le principe de SEARN et les garanties que présente cette méthode d'apprentissage [5]. Nous supposerons, dans la suite, avoir accès à une politique oracle π^{oracle} capable de déterminer, parmi toutes les actions réalisables dans un état donné, l'action (supposée unique) permettant d'obtenir la solution avec la meilleure récompense cumulée.

SEARN (Daumé III *et al.*, 2009) est un algorithme générique d'apprentissage par imitation utilisé pour apprendre les paramètres d'une politique gloutonne reposant sur la règle de décision décrite par l'équation (2). L'idée centrale de SEARN est de construire l'ensemble d'apprentissage (les états à parcourir, les actions à effectuer) de manière itérative en commençant par suivre la distribution des états engendrée par la politique oracle et en introduisant progressivement les états engendrés par la politique apprise. Ce processus permet de limiter le nombre d'états considérés lors de l'apprentissage tout en garantissant que les exemples sur lesquels la politique est apprise seront similaires aux cas qui seront vus en test : si les exemples n'étaient engendrées qu'à partir de la politique oracle, le système ne saurait pas quelle décision prendre dès qu'il s'en éloignerait puisqu'il n'aurait jamais été confronté à une telle situation en apprentissage.

Plus précisement, comme le montre l'algorithme 1, SEARN apprend une suite de politiques $\pi^0, \pi^1, \ldots, \pi^N$ où N, le nombre d'itérations, est un hyper-paramètre de l'algorithme. La politique initiale π^0 est la politique oracle π^{oracle} ; puis, à la i^{e} itération, la politique courante est utilisée pour engendrer un nouvel ensemble d'apprentissage : pour chaque séquence d'observations du corpus, π^i est utilisée pour prédire une séquence d'étiquettes et π^{oracle} pour déterminer la bonne action [6] qui aurait dû être prise dans chaque état visité. Concrètement, dans chaque état visité, on ajoute à l'ensemble d'apprentissage en cours de construction un exemple qui associe à l'état l'action qu'aurait pris l'oracle. À la fin de l'itération, les paramètres d'une politique ρ^{i+1} sont estimés sur cet ensemble d'apprentissage à l'aide d'un algorithme de

4. Lorsque les exemples sont i.i.d., la fonction qui minimise, à l'intérieur d'une classe de fonctions donnée, l'erreur sur l'ensemble d'apprentissage est aussi celle qui minimise l'erreur sur l'ensemble de test avec une forte probabilité.

5. Par souci de simplification, nous avons présenté SEARN dans le cas où la contribution d'une action à la fonction de coût est soit 0 (lorsque la prédiction résultant de l'action est correcte), soit 1 (lorsque cette prédiction est erronée). Cette situation correspond, par exemple, à une évaluation par une distance de Hamming. La généralisation à d'autres fonctions de coût nécessite est possible en considérant une généralisation de la classification multi-classes, la classification multi-classes *pondérée (cost-sensitive classification)* qui ordonne les erreurs selon leur « gravité ». Cette généralisation ne change pas substantiellement l'esprit de l'algorithme, ni notre discussion.

6. La bonne action à prendre est choisie de cette manière dans SMILE, qui est la version de SEARN présentée dans (Ross & Bagnell, 2010).

classification multi-classes standard :

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{s,a \in \mathcal{S}} \mathbb{1} \left\{ \arg\max_{a'} \langle \phi(s, a') | \mathbf{w} \rangle = a \right\} \tag{4}$$

La $(i+1)^{\text{e}}$ politique π^{i+1} est alors définie comme un *mélange stochastique* de la politique π^i et de la politique ρ^{i+1} :

$$a = \pi^{i+1}(s) = \begin{cases} \rho^{i+1}(s) & \text{avec une probabilité } \beta \\ \pi^i(s) & \text{avec une probabilité } 1-\beta \end{cases} \tag{5}$$

où β, la probabilité de choisir la politique que l'on vient d'apprendre, est le second hyper-paramètre de l'algorithme. Ce mélange stochastique permet de contrôler la vitesse à laquelle on s'éloigne de la politique oracle : la probabilité d'appliquer π^{oracle} à la i$^{\text{e}}$ itération est égale à $(1-\beta)^i$.

Algorithme 1 : Principe de SEARN / SMILE

Entrées : un ensemble de séquences étiquetées $\mathcal{T} = (\mathbf{x}_i, \mathbf{y}_i)_{i=1}^n$, un nombre d'itérations N, $\beta \in [0,1]$ vitesse à laquelle on s'éloigne de la politique oracle

Initialiser π^0;
for $i = 0 : N$ **do**
  $S \leftarrow \emptyset$; ▷ Ensemble des exemples collectés
  for $\mathbf{x}, \mathbf{y} \in \mathcal{T}$; ▷ Décodage avec la politique π^{i-1}
  do
    $\langle \mathcal{S}, \mathcal{A}, \mathcal{S}_f, s_0 \rangle \leftarrow$ search_space($\mathbf{x}$) ; ▷ Crée le PDM associé à l'exemple
    $s \leftarrow s_0$; ▷ État courant
    while $s \notin \mathcal{S}_f$ **do**
      $S \leftarrow S \cup \{(s, \pi^{\text{oracle}}(s))\}$;
      $s = \pi^i(s)$; ▷ Passe à l'état suivant
    end
  end
  apprendre une nouvelle politique ρ^{i+1} sur S;
  $\pi^{i+1} = (1-\beta) \cdot \pi^i + \beta \cdot \rho^{i+1}$; ▷ Mélange stochastique
end

Sortie : $\tilde{\pi}^N$

À la fin de l'apprentissage, le résultat de l'algorithme est la politique $\tilde{\pi}^N$ qui s'obtient en remplaçant dans la politique π^N la composante correspondant à la politique initiale π_0 (la politique oracle) par la politique qui choisit au hasard une des actions accessibles dans un état. En effet, la politique oracle n'est connue que pour les données d'entraînement ; la politique $\tilde{\pi}^N$ pourra être appliquée à des séquences d'observations dont les étiquettes de référence sont inconnues.

La version présentée ici de SEARN possède des garanties théoriques (Ross & Bagnell, 2010) bornant l'erreur en généralisation du classifieur structuré qui reposent sur deux caractéristiques de l'algorithme. Premièrement, la première politique appliquée est la politique oracle ; le premier classifieur appris permet alors de retrouver la meilleure séquence d'actions. Deuxièmement, grâce à l'utilisation du mélange stochastique, les états parcourus lors du décodage changent lentement, ce qui permet de limiter l'effet de propagation d'erreur lors du passage d'une politique π^i à la politique π^{i+1}.

3.3 Détermination de la politique oracle

La mise en œuvre de SEARN pour l'apprentissage de modèles de séquences suppose que l'on ait accès à une politique oracle capable de déterminer quelle action doit être effectuée dans un état donné. Déterminer la politique oracle lorsque l'on considère le modèle d'étiquetage gauche-droite et la distance de Hamming comme fonction de coût est aisé : à chaque état, l'oracle renvoie l'action qui associe la bonne étiquette (selon la référence) à la position suivante.

Par contre, dans le cas d'un décodage en ordre libre (toujours évalué par une distance de Hamming), plusieurs séquences d'actions peuvent engendrer la séquence d'étiquettes de référence et, à chaque état, plusieurs actions pourront être considérées comme optimales : quelle que soit la fonction de coût considérée, il y a toujours autant d'actions optimales que de positions non-étiquetées. Cela est directement dû au fait que l'ordre dans lequel la séquence d'étiquettes est générée est

une variable cachée : seul l'étiquetage final est observé. L'oracle sera donc une fonction de $\mathcal{S}$ dans $\wp(\mathcal{A}_s)$ et renverra non plus une unique action optimale mais un ensemble d'actions optimales. Suivant Goldberg & Nivre (2013), nous qualifions ces oracles de *non-déterministes*.

La prise en compte d'oracles non-déterministes dans SEARN nécessite d'adapter l'algorithme d'apprentissage : il n'est, en effet, plus possible de réduire directement le problème de prédiction structuré à un problème de classification multi-classes, puisqu'il n'y a plus dans chaque état une unique bonne réponse. Deux stratégies peuvent être envisagées pour résoudre ce problème.

Suppression de l'ambiguïté La première stratégie consiste à supprimer l'ambiguïté en choisissant, parmi toutes les actions optimales identifiées par l'oracle, $\pi^{\text{oracle}}(s)$, une étiquette $\hat{a}$ qui sera considérée comme l'étiquette devant être prédite et vers laquelle devra être faite une éventuelle mise à jour du vecteur de paramètres. C'est également cette action qui déterminera dans quel état ira le système (c.-à-d. celle qui sera utilisée pour effectuer la ligne 10 de l'algorithme 1).

Cette stratégie a été proposée par (Shen *et al.*, 2007) dans un contexte similaire [7] et a été reprise, entre autres, dans (Goldberg & Nivre, 2012; Ma *et al.*, 2012; Gesmundo & Henderson, 2014). Shen *et al.* (2007) préconisent de prendre comme action de référence, l'action optimale qui possède le plus grand score :

$$\hat{a} = \underset{a \in \pi^{\text{oracle}}(s)}{\arg\max}\ f(s,a) \tag{6}$$

En effet, intuitivement, cette action correspond à l'action optimale dont la prédiction nécessitera la plus petite mise à jour et donc la plus petite remise en cause du vecteur de paramètres courant.

Le principal avantage de cette stratégie est de ne pas nécessiter de modification de l'algorithme d'apprentissage. Mais les hypothèses sur lesquelles reposent les garanties théoriques de SEARN sont violées. En particulier, il n'est plus possible, dans cette stratégie de contrôler la vitesse à laquelle on s'éloigne de la politique optimale.

Réduction à un problème d'ordonnancement Nous introduisons, dans ce travail, une stratégie alternative qui consiste à considérer toutes les actions optimales identifiées par l'oracle comme étant correctes et, en cas d'erreur, à renforcer chacune de ces actions.

Plus précisement, nous proposons de réduire le problème de prédiction structuré, non plus à un problème de classification multi-classes, mais à un problème d'ordonnancement bipartite (Liu, 2009) dont l'objectif est de distinguer un ensemble d'exemples positifs d'un ensemble d'exemples négatifs en assurant que tous les exemples positifs ont un score plus grand que tous les éléments négatifs. La collecte par SEARN des exemples d'apprentissage (ligne 9 de l'algorithme 1) correspond alors à l'opération :

$$S \leftarrow S \cup \left\{(s,a), \forall a \in \pi^{\text{oracle}}(s)\right\} \tag{7}$$

Lors du décodage, l'état suivant est déterminé en exécutant aléatoirement une des actions de $\pi^{\text{oracle}}(s)$. On peut montrer qu'avec cette stratégie, les garanties théoriques de SEARN restent valables.

Une comparaison expérimentale de ces deux stratégies est présentée à la section 4.

4 Expériences

Dans cette section, nous étudions les performances des deux modèles introduits dans ce travail, le modèle gauche-droite et le modèle ordre libre, sur différentes tâches de TAL. Nous commencerons par présenter notre protocole expérimental avant de rapporter et discuter nos résultats.

4.1 Protocole expérimental

Dans toutes nos expériences nous utilisons, comme classifieur multi-classes, une machine à vecteurs supports (SVM) avec un noyaux linéaire et une régularisation L2 ; la valeur du paramètre contrôlant la régularisation est systématiquement

7. (Shen *et al.*, 2007) introduit cette idée pour l'apprentissage d'un analyseur morpho-syntaxique *en ligne* ; dans ce travail, nous ne considérons que des méthodes d'apprentissage *batch*.

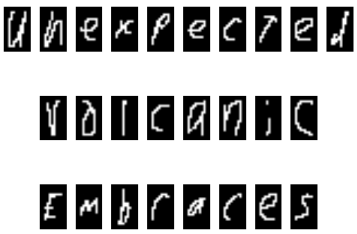

FIGURE 3 – Exemple de données pré-segmentées pour la reconnaissanse de l'écriture manuscrite.

aerodrome	E-rxdrom-	$1- < 0 >> 2 < -$
aeronaut	E-rxnc-t	$1- < 0 > 2- <$
aeronautics	E-rxnc-tIks	$2- < 0 > 1- < 0 <<$
aeroplane	E-rxplen-	$1- < 0 >> 2 < -$

FIGURE 4 – Extrait de NETTALK : à chaque mot est associé une séquence de phonèmes (avec '-' un symbole « NULL », et structure prosodique (0,1,2 marquent differents degrés d'accentuation pour les voyelles, > et < marquent respectivement les attaques et coda de syllabes).

déterminée par validation croisée. Les résultats de nos deux modèles sont comparés à un SVM [8] multi-classes « simple » n'utilisant aucune information sur les étiquettes du voisinage et à un CRF linéaire [9] considérant uniquement l'étiquette précédente comme information de structure et réalisant une recherche exacte de la solution optimale.

Ces méthodes d'étiquetage de séquences sont comparées sur quatre tâches différentes d'étiquetage de séquences, qui ont été choisies car elles mettent en jeu un grand nombre d'étiquettes, ce qui rend l'apprentissage et l'inférence computationnellement coûteux :

Reconnaissance de l'écriture manuscrite Le corpus [10] utilisé contient 44 images de 150 mots (soit 6 600 exemples au total). Il s'agit d'un corpus artificiel, très structuré (la plupart des combinaisons d'étiquettes sont interdites et la connaissance d'une étiquette désambiguïse fortement les lettres voisines) qui est généralement utilisé pour tester les méthodes d'apprentissage structuré.

Chaque image est pré-segmentée en séquence d'images de lettres de taille 8×16 pixels. Quelques exemples de données pré-segmentées sont représentées sur la Figure 3. Les données sont réparties en 10 paquets ; nous utilisons 9 paquets pour l'entraînement et 1 paquet pour le test et nous faisons la validation croisée sur les 10 configurations.

Nous utilisons, comme caractéristiques, la valeur des 144 pixels ainsi que les 9 étiquettes précédentes.

Prononciation automatique L'objectif de cette tâche est de déterminer automatiquement la prononciation d'un mot à partir de la suite de lettres le composant. L'information de prononciation est constituée de deux parties qui seront apprises et évaluées séparément : une suite de phonèmes et une description de la structure prosodique.

Dans nos expériences, nous utilisons le corpus NETTALK [11] (Sejnowski & Rosenberg, 1987), qui contient 20 008 mots anglais accompagnés d'une information de prononciation. Pour chaque mot anglais contenant T lettres, la représentation phonologique correspondante est encodées par deux séquences de T symboles : une séquence de phonèmes, utilisant un alphabet de 51 étiquettes (dont un symbole NULL) et une séquence décrivant la structure prosodique (pour les consonnes, la position dans la syllabe, pour les voyelles le degré d'accentuation) avec un alphabet de 6 symboles (dont un symbole NULL). Un extrait de NETTALK est représenté sur la Figure 4. Les données sont partitionnées en 10 paquets. Nous utilisons 8 paquets pour l'entraînement, 1 paquet pour l'optimisation de l'hyper-paramètre et 1 paquet pour le test.

Nous utilisons comme caractéristiques les 4-grammes de lettres dans une fenêtre de taille ± 9 par rapport à la position courante. Les caractéristiques de structure correspondent aux étiquettes décodées (phonèmes ou marques prosodiques) ainsi que leurs combinaisons avec les caractéristiques de base.

Analyse morpho syntaxique de l'allemand Pour cette tâche nous avons utilisé le corpus TIGER [12], contenant 50 000

8. Nous avons utilisé l'implémentation des SVM fournie dans la bibliothèque SCIKIT-LEARN (Pedregosa *et al.*, 2011)
9. Implémenté dans Wapiti (Lavergne *et al.*, 2010) : `http://wapiti.limsi.fr`
10. `http://www.seas.upenn.edu/~taskar/ocr/`
11. `https://archive.ics.uci.edu/ml/datasets/Connectionist+Bench+(Nettalk+Corpus)`
12. `http://www.ims.uni-stuttgart.de/forschung/ressourcen/korpora/tiger.en.html`

	Écriture	Phonèmes	Accents	POS tags
SVM non-struct	74,97%	93,17%	91,49%	96,34%
Gauche-droite	91,93%	92,51%	90,78%	96,48%
Ordre libre	97,28%	93,32%	92,28%	96,91%
CRF	90,28%	93,02%	91,71%	97,00%
Gauche-droite (sans prop.)	96,56%	94,41%	94,31%	96,73%
Ordre libre (sans prop.)	98,65%	94,10%	94,12%	97,11%

TABLE 1 – Performances (% d'étiquettes correctes) des différents apprentis considérés sur 4 tâches standard.

phases allemandes, soit 900 000 mots étiquetés avec leur catégorie syntaxique. Ce corpus distingue 54 catégories différentes. Nous utilisons le partitionnement standard de ce corpus (80% des données pour l'entraînement, 10% pour la validation et 10% pour le test).

Dans nos expériences nous utilisons les caractéristiques suivantes : des informations de surface du mot courant (présence de majuscules, de chiffres, ...), les préfixes et les suffixes de taille de 1 à 4, les mots dans une fenêtre de taille 2 par rapport au mot considéré. Le mot courant est également combiné avec les 9 étiquettes précédentes.

4.2 Mise-en-œuvre de SEARN

Afin de réduire la complexité de l'apprentissage, nous avons nous avons, dans notre implémentation de SEARN, simplifié la définition du mélange stochastique défini par l'équation (5) en ne considérant que les deux dernières politiques apprises. Formellement, nous supposons que $\forall 0 < j < i, \rho^j = \rho^{i-1}$. La politique stochastique à l'itération i mélange donc simplement le dernier classifieur appris ρ^{i-1} (avec probabilité $1-(1-\beta)^i$) et la politique optimale π_0. Par conséquent, la politique finale $\tilde{\pi}^N$ ne comporte plus qu'une composante ρ^N.

Cette approximation s'appuie sur le fait que les classifieurs les plus probables sont les classifieurs les plus récents, qui diffèrent peu du dernier classifieur appris. Des expériences préliminaires ont montré que cette simplification n'avait pas d'impact sur les performances de l'approche.

Suivant Daumé III *et al.* (2009), nous avons fixé β à $\frac{1}{T}$ où T est la longueur de la phrase considérée. Considérer des valeurs de β plus petite n'améliore pas les performances mais augmente significativement la vitesse de convergence ; des valeurs plus grandes de β dégradent les performances.

Sauf mention contraire, dans toutes nos expériences, nous avons utilisé pour déterminer la politique oracle la stratégie consistant à conserver l'ambiguïté (la seconde des stratégies présentées à la section 3.3). En effet, celle-ci a obtenu de meilleurs résultats dans nos expériences préliminaires.

4.3 Résultats et discussion

Les principaux résultats expérimentaux sont résumés dans la Table 1.

Ces résultats montrent, conformément à l'intuition, que la recherche exacte (mise en œuvre uniquement dans le CRF) améliore, légèrement, les résultats par rapport à une approche gloutonne sauf quand les sorties sont très structurées (comme pour la tâche de reconnaissance de l'écriture). En effet, dans ce cas la connaissance d'un historique riche (les 9 dernières étiquettes prédites alors que le CRF ne considère des dépendances qu'entre deux étiquettes consécutives) apporte une information suffisante pour compenser les erreurs induites par l'algorithme de recherche approchée. Ce phénomène apparaît également lorsque l'on considère un espace de recherche plus grand : le modèle « ordre libre » obtient des performances au moins aussi bonnes que le CRF et arrive même souvent à obtenir des résultats meilleurs.

Pour évaluer la capacité du modèle ordre libre à s'éloigner du décodage monotone, nous avons calculé la différence moyenne entre l'indice de la décision et la position concernée dans le mot. Pour la tâche de prosodie, cette valeur est de 2,7 ; pour la tâche de reconnaissance de phonèmes, elle est de 2,9. Ces deux observations montrent que l'ordre du décodage est, en pratique, assez éloigné de l'ordre monotone.

Une étude qualitative de l'ordre dans lequel le système effectue les actions, montre que la « facilité » de l'action correspond souvent à notre intuition. Par exemple, dans le cadre de la tâche de prosodie, on peut imaginer que l'accent secondaire

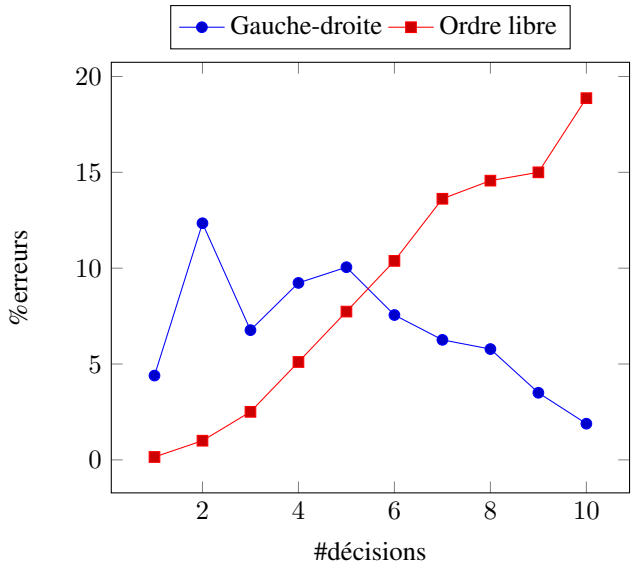

FIGURE 5 – Conversion graphème-phonème. Taux d'erreur en fonction de l'indice de la décision : modèle gauche-droite (à gauche) et modèle ordre libre (à droite)

peut être plus facile à détecter quand on connaît déjà l'information relative à l'accent primaire. En analysant l'ordre d'attribution des étiquettes, on peut voir que l'accent secondaire (quand il existe) dans un mot est mis après l'accent primaire dans 70% des cas. De manière générale, le système a tendance à décoder les consonnes (sur lesquels on commet 31% du nombre total d'erreurs) avant les voyelles (62% de toutes les erreurs) ; le reste des erreurs est commis sur les sons muets (7%).

Pour illustrer la capacité du modèle « ordre libre » à commencer par choisir les étiquettes les plus faciles en premier, nous avons représenté, à la figure 5, le taux d'erreur en fonction de l'indice de la décision dans la séquence pour la tâche graphème-phonème. Cette figure montre clairement que, dans le modèle « ordre libre », les erreurs apparaissent beaucoup plus tardivement lors du décodage, ce qui limite le problème de la propagation d'erreurs et permet d'obtenir les bonnes performances observées.

Pour quantifier l'effet de la propagation d'erreur pour les méthodes d'apprentissage par imitation, nous avons effectué l'expérience de contrôle suivante : le décodage incrémental est toujours effectué par la politique apprise (et les erreurs de prédictions ont toujours lieu), mais l'historique est systématiquement remis à jour avec l'étiquette de référence (et ne contient donc aucune erreur). Les résultats pour les différentes tâches, présentés dans la seconde partie de la Table 1, montre que l'impact de la propagation des erreurs (entre 0,3 et 5 points suivant les tâches) est loin d'être négligeable et la propagation des erreurs doit effectivement être contrôlée.

Les résultats présentés Table 1 ont été obtenus avec l'oracle conservant l'ambiguïté. Les performances obtenues par l'oracle levant l'ambiguïté sont nettement moins bonnes : ce dernier obtient un score de 90,01% (-2,27%) sur la tâche de prédiction de la structure prosodique et un score de 91,80% (-1,52%) sur la tâche de reconnaissance des phonèmes.

5 Conclusion

Nous avons présenté dans ce travail un nouveau formalisme pour la prédiction d'objets structurés, comme les arbres et les séquences. Ce formalisme, qui consiste à voir l'apprentissage structuré comme un processus incrémental au cours duquel la sortie est progressivement construite, permet d'inscrire l'apprentissage structuré dans le cadre de l'apprentissage par renforcement. Grâce à ce lien, nous avons pu introduire une méthode théoriquement bien justifiée pour apprendre des méthodes d'inférence approchée et ainsi proposer des méthodes d'étiquetage de séquences rapides et capables de prendre en compte des dépendances riches. Cette approche est validée par des résultats équivalents ou supérieurs aux méthodes état de l'art sur quatre tâches variées du TAL.

L'apprentissage structuré incrémental est un cadre d'apprentissage général qui n'est pas limité à la prédiction de séquences

et nous envisageons d'appliquer les méthodes d'apprentissage décrites dans ce travail à des problèmes plus complexes comme l'analyse en dépendances ou l'apprentissage des modèles de séquences factorisés capables de prédire, de manière jointes, plusieurs étiquettes pour chaque observations. Nous envisageons également d'approfondir les résultats théoriques offertes par les méthodes d'apprentissage incrémental afin de mieux comprendre les principes sur lesquels les méthodes de prédiction *easy first*.

Références

ABBEEL P. & NG A. Y. (2004). Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, p. 1–, New York, NY, USA : ACM.

CHIANG D. (2007). Hierarchical phrase-based translation. *Computational Linguistics*, **33**(2), 201–228.

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.

DAUMÉ III H., LANGFORD J. & MARCU D. (2009). Search-based structured prediction. *Machine Learning Journal*.

DAUMÉ III H. & MARCU D. (2005). Learning as search optimization : approximate large margin methods for structured prediction. In *ICML '05 : Proceedings of the 22nd international conference on Machine learning*, p. 169–176, New York, NY, USA : ACM Press.

FINKEL J. R., MANNING C. D. & NG A. Y. (2006). Solving the problem of cascading errors : Approximate bayesian inference for linguistic annotation pipelines. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, p. 618–626, Sydney, Australia.

GESMUNDO A. & HENDERSON J. (2014). Undirected machine translation with discriminative reinforcement learning. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, p. 10–19, Gothenburg, Sweden : Association for Computational Linguistics.

GOLDBERG Y. & ELHADAD M. (2010). An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies : The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, p. 742–750.

GOLDBERG Y. & NIVRE J. (2012). A dynamic oracle for arc-eager dependency parsing. In *Proceedings of COLING 2012*, p. 959–976 : The COLING 2012 Organizing Committee.

GOLDBERG Y. & NIVRE J. (2013). Training deterministic parsers with non-deterministic oracles. *Transactions of the Association of Computational Linguistics*, **1**, 403–414.

LANGFORD J. & ZADROZNY B. (2005). Relating reinforcement learning performance to classification performance. In *Proceedings of the International Conference on Machine Learning (ICML)*.

LAVERGNE T., CAPPÉ O. & YVON F. (2010). Practical very large scale CRFs. In *Proceedings the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, p. 504–513.

LIU T.-Y. (2009). Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, **3**(3), 225–331.

MA J., XIAO T., ZHU J. & REN F. (2012). Easy-first chinese pos tagging and dependency parsing. In *Proceedings of COLING 2012*, p. 1731–1746 : The COLING 2012 Organizing Committee.

PEDREGOSA F., VAROQUAUX G., GRAMFORT A., MICHEL V., THIRION B., GRISEL O., BLONDEL M., PRETTENHOFER P., WEISS R., DUBOURG V., VANDERPLAS J., PASSOS A., COURNAPEAU D., BRUCHER M., PERROT M. & DUCHESNAY E. (2011). Scikit-learn : Machine learning in Python. *Journal of Machine Learning Research*, **12**, 2825–2830.

ROSS S. & BAGNELL J. A. D. (2010). Efficient reductions for imitation learning. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*.

SEJNOWSKI T. J. & ROSENBERG C. R. (1987). Parallel networks that learn to pronounce English text. *Complex Systems*, **1**, 145–168.

SHEN L., SATTA G. & JOSHI A. (2007). Guided learning for bidirectional sequence classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, p. 760–767, Prague, Czech Republic.

SUTTON R. & BARTO A. (1998). *Reinforcement learning : an introduction*. MIT Press.

YAMADA H. & MATSUMOTO Y. (2003). Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*.