Étude de réseaux de neurones récurrents pour l'étiquetage de séquences

Marco Dinarelli Isabelle Tellier
LaTTiCe (UMR 8094), CNRS, ENS Paris, Université Sorbonne Nouvelle - Paris 3
1 rue Maurice Arnoux, 92120 Montrouge, France
PSL Research University, USPC (Université Sorbonne Paris Cité)
marco.dinarelli@ens.fr, isabelle.tellier@univ-paris3.fr

RÉSUMÉ

Dans cet article nous étudions plusieurs types de réseaux neuronaux récurrents (RNN) pour l'étiquetage de séquences. Nous proposons deux nouvelles variantes de RNN et nous les comparons aux variantes plus classiques de type Jordan et Elman. Nous expliquons en détails quels sont les avantages de nos nouvelles variantes par rapport aux autres RNN. Nous évaluons tous les modèles, les nouvelles variantes ainsi que les RNN existants, sur deux tâches de compréhension de la parole : ATIS et MEDIA. Les résultats montrent que nos nouvelles variantes de RNN sont plus efficaces que les autres.

ABSTRACT

A study of Recurrent Neural Networks for Sequence Labelling

In this paper we study different types of Recurrent Neural Networks (RNNs) for sequence labelling tasks. We propose two new types of RNNs and we compare them to the more traditional Elman and Jordan RNNs. We compare all models on two different tasks of sequence labelling, namely ATIS and MEDIA. The results show that the new variants of RNNs are more effective than the others.

MOTS-CLÉS : Réseaux de neurones récurrents, étiquetage de séquences, compréhension de la parole.

KEYWORDS: Recurrent Neural Networks, Sequence Labelling, Spoken Language Understanding.

1 Introduction

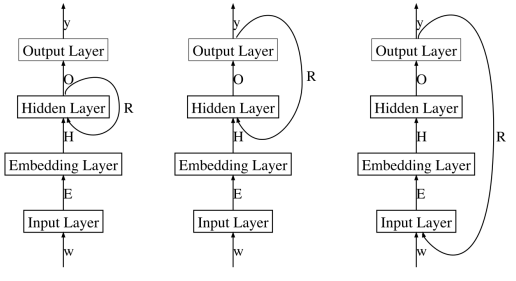
Les réseaux de neurones récurrents (RNN) (Jordan, 1989; Elman, 1990) sont des modèles capables de prendre en compte un *contexte* dans leur fonction de décision. Ils sont pour cela particulièrement adaptés à plusieurs tâches de Traitement Automatique des Langues (TAL), notamment celles qui consistent à prédire une information séquentielle (Collobert & Weston, 2008; Collobert *et al.*, 2011; Yao *et al.*, 2013; Mesnil *et al.*, 2013; Vukotic *et al.*, 2015). La plupart des modèles probabilistes utilisés pour ce type de tâches (même les tout premiers, historiquement) sont aussi capables d'utiliser un contexte dans leur fonction de décision. Dans le cas des RNN, l'information contextuelle est donnée par une connexion en boucle. Cette connexion permet de prendre en compte à l'étape courante une ou plusieurs informations prédites dans une étape précédente. Cette architecture semble d'autant plus efficace dans les réseaux neuronaux qu'elle peut se combiner avec la puissance des représentations distributionnelles (dites aussi "plongements" ou *embeddings* en anglais).

Dans la littérature sur les réseaux neuronaux récurrents pour le TAL, deux types principaux d'architecture ont été proposés, appelés aussi RNN *simples* : les RNN d'Elman (Elman, 1990) et ceux de Jordan (Jordan, 1989). La différence entre ces deux modèles réside simplement dans la connexion qui donne son caractère récurrent au réseau de neurones : dans les RNN d'Elman, la boucle se situe au

niveau de la couche cachée, alors qu'elle se trouve entre la couche de sortie et la couche cachée dans les RNN de Jordan. Dans ce dernier cas, elle permet d'utiliser à l'étape courante les étiquettes prédites pour des positions précédentes d'une séquence. Ces dernières années, ces deux types de RNN ont eu beaucoup de succès, notamment pour créer des modèles de langues (Mikolov *et al.*, 2010, 2011), ou pour certaines tâches d'étiquetage de séquences (Yao *et al.*, 2013; Mesnil *et al.*, 2013; Vukotic *et al.*, 2015).

L'intuition à la base de cet article est que les plongements peuvent permettre une modélisation plus fine et efficace non seulement des mots, mais également des *étiquettes* et de leurs dépendances, qui sont cruciales dans les tâches d'étiquetage de séquence. Dans cet article, nous définissons deux nouvelles variantes de RNN favorisant cette modélisation. Dans la première, la connexion récurrente relie la couche de sortie et la couche d'entrée. Cette variante redonne donc en entrée du réseau les étiquettes prédites aux étapes précédentes du traitement d'une séquence, en plus de l'entrée habituelle constituée d'un ou plusieurs mots. D'après notre intuition et grâce à l'utilisation de plongements sur les étiquettes, cette variante modélise de façon plus efficace les dépendances entre étiquettes. La seconde variante envisagée est une hybridation de la première avec les réseaux d'Elman.

Un schéma général des réseaux d'Elman, de Jordan et de la variante principale proposée ici est montré dans la Figure 1. Dans cette figure, w sont les mots, y les étiquettes, et E, H, O et R sont les paramètres du modèle. Ils seront décrits en détails dans la prochaine section. Mais, avant cela, nous allons justifier l'intérêt de notre proposition en expliquant en quoi la modification (apparemment simple) introduite par cette nouvelle variante offre plusieurs avantages par rapport aux architectures plus classiques d'Elman et de Jordan (Elman, 1990; Jordan, 1989).



(a) RNN d'Elman (b) RNN de Jordan (c) Notre variante de RNN

FIGURE 1 – Schéma général des principaux réseaux neuronaux utilisés dans cet article.

Tout d'abord, puisque la sortie des étapes précédentes est redonnée en entrée à l'étape courante, l'information contextuelle reparcourt le réseau dans sa totalité, affectant ainsi toutes les couches du réseau, et cela quel que soit le mode de propagation (en avant ou en arrière). Cela permet un apprentissage plus efficace des paramètres du réseau. Dans les architectures d'Elman et de Jordan, seulement une partie des couches est affectée par l'information contextuelle.

Un autre avantage de cette nouvelle variante est lié aux plongements des unités de sortie. En effet, la première couche du réseau est une table de "look-up" qui sert à transformer la représentation "one-hot" [1] très creuse des unités en entrée en représentation distributionnelle. Puisque, dans notre nouvelle variante, la sortie est re-donnée en entrée au réseau, le changement de représentation (de creuse à distributionnelle) vaut pour les mots aussi bien que pour les étiquettes. Les représentations distributionnelles des étiquettes peuvent donc être pré-apprises, comme elles le sont déjà pour les mots.

1. La représentation "one-hot" d'un élément en position i d'un dictionnaire V est un vecteur de taille $|V|$ dont la $i$ème coordonnée est égale à 1 et toutes les autres sont à 0

Les plongements de mots fournis par *word2vec* ont déjà largement montré leur intérêt syntaxique et sémantique pour diverses applications (Mikolov *et al.*, 2013a,b).

Un troisième avantage est le fait que les représentations distributionnelles d'étiquettes transmettent dans le réseau une "quantité information" plus grande que celle transmise par les connexions récurrentes des réseaux d'Elman et de Jordan. Dans le réseau d'Elman, la connexion récurrente est une boucle dans la couche cachée, elle n'affecte donc pas la couche d'entrée dans la phase de propagation en avant. De plus, dans ce réseau, on n'utilise pas explicitement l'information donnée par la couche de sortie. Dans le réseau de Jordan, en revanche, on utilise la sortie en la re-donnant en entrée à la couche cachée. Cependant, la représentation de la sortie dans un tel réseau est soit une distribution de probabilités sur les étiquettes, soit une représentation creuse "one-hot" de l'étiquette prédite. Il est clair que la représentation "one-hot" ne transmet pas beaucoup d'information au réseau. La distribution de probabilités devrait en fournir plus, puisqu'elle représente une décision "smooth" (lissée). Cependant, d'après nos expériences, cette distribution est très biaisée en faveur d'une ou d'un petit nombre d'étiquettes, ne fournissant donc en fin de compte pas beaucoup plus d'information qu'une représentation creuse "one-hot". Ce biais est déjà évoqué dans (Mesnil *et al.*, 2013), où les auteurs montrent que le réseau de Jordan est plus performant quand il est appris sur les représentations creuses des étiquettes de référence que sur les représentations creuses des étiquettes prédites et sur les distributions de probabilités prédites. La différence entre le RNN de Jordan et notre variante est un point clef de notre proposition, une comparaison plus détaillée est donnée plus loin.

Un dernier avantage de la connexion récurrente entre la couche de sortie et celle d'entrée est la robustesse qu'elle apporte. Les représentations distributionnelles des étiquettes rendent le modèle plus robuste vis-à-vis des erreurs de prédiction qui peuvent survenir pendant le traitement d'une séquence. Le modèle de Jordan, avec des représentations creuses ou des distributions de probabilités biaisées de la sortie, est plus vulnérable à la propagation des erreurs.

Dans cet article, tous les RNN sont étudiés dans leur version "en avant" (*forward* par la suite), "en arrière" (*backward*) et bidirectionnelle (Schuster & Paliwal, 1997). Nous avons utilisé notre propre implémentation des réseaux d'Elman et de Jordan, afin que la comparaison avec nos variantes soit directe et plus fiable. Nous évaluons tous les modèles sur deux tâches différentes de compréhension de la parole (De Mori *et al.*, 2008) : ATIS (Dahl *et al.*, 1994) et MEDIA (Bonneau-Maynard *et al.*, 2006), qui peuvent toutes les deux être ramenées à un étiquetage de séquences. ATIS est relativement simple et ne nécessite pas une modélisation poussée des dépendances entre étiquettes. MEDIA est une tâche plus difficile pour laquelle la capacité d'un modèle à prendre en compte les dépendances entre étiquettes est cruciale.

Le reste de l'article est organisé comme suit. Dans la section suivante, nous décrivons en détails les réseaux neuronaux récurrents, partant des modèles classiques d'Elman et de Jordan pour arriver à nos deux variantes. La section 3 est consacrée à la présentation des corpus sur lesquels nous évaluons les modèles, aux paramétrages choisis et aux résultats obtenus. La section 4 conclut la présentation.

2 Réseaux neuronaux récurrents

2.1 Fonctionnement général

Les réseaux de neurones récurrents évoqués dans cet article ont la même architecture que le modèle de langage neuronal (NNLM de l'anglais *Neural Network Language Model*) décrit dans (Bengio *et al.*, 2003). Cette architecture comprend 4 couches : la couche d'entrée, des plongements, une couche cachée et une de sortie. Les mots sont donnés en entrée au réseau sous la forme d'un index qui correspond à leur position dans un dictionnaire V.

L'index d'un mot est utilisé pour sélectionner sa représentation distributionnelle (*plongement* par la suite) dans une matrice $E \in R^{|V| \times N}$, $|V|$ étant la taille du dictionnaire et N la taille des plongements

(N est un hyper-paramètre du modèle, à choisir). Comme le montre la Figure 1, la matrice E contient l'ensemble des paramètres, ou poids, entre la couche d'entrée et la couche des plongements. La notation $E(v(w_t))$ désigne le plongement du mot w donné en entrée au réseau à l'étape t du traitement d'une séquence. $v(w_t) = i$ est l'index du mot w_t dans le dictionnaire. Par raccourci d'écriture, nous utilisons $v(w_t)$ pour désigner aussi la représentation creuse "one-hot" du mot w_t, c'est-à-dire un vecteur numérique de taille $|V|$ qui vaut 0 partout sauf à la position $v(w_t)$. Puisque, dans cet article, nous traitons de l'étiquetage de séquences, l'étape t, ou temps t, dans le traitement d'une séquence, correspond en fait à la position t dans la séquence. [2]

Nous notons respectivement H et O les matrices à valeurs réelles des poids entre la couche des plongements et la couche cachée d'une part, entre la couche cachée et la couche de sortie d'autre part. La sortie d'un réseau, étant donné le mot w_t en entrée, est y_t. La fonction d'activation que nous utilisons dans la couche cachée de tous les réseaux est la sigmoïde [3]. La couche de sortie est un *Softmax* [4]. La sortie du réseau de neurones y est donc en fait une distribution de probabilités sur l'ensemble des étiquettes L à prédire.

Par rapport aux NNLM, les RNN ont en plus une connexion récurrente entre deux couches qui dépend du type de RNN en question : comme nous l'avons mentionné dans l'introduction, le RNN d'Elman (Elman, 1990) a une boucle au niveau de la couche cachée. Puisque la couche cachée encode la représentation interne de l'information en entrée, la boucle d'un réseau d'Elman permet de garder "en memoire" les mots utilisés en entrées aux positions précédentes dans le traitement d'une séquence. En revanche le RNN de Jordan (Jordan, 1989) a une connexion récurrente entre la couche de sortie et la couche cachée. Pour prédire l'étiquette du mot à la position courante, un réseau de Jordan peut donc prendre en compte les étiquettes prédites aux positions précédentes. Quel que soit le type de RNN en question, nous notons R la matrice de poids associée à la connexion récurrente.

Pour le calcul de la sortie des différentes couches, nos réseaux d'Elman et de Jordan sont standards et nous renvoyons donc à la littérature qui les décrit en détails (Elman, 1990; Jordan, 1989; Bengio, 2012; Mesnil *et al.*, 2013)

Dans les modèles de TAL pour l'annotation, il est commun d'utiliser un contexte en entrée. On peut utiliser comme contexte une fenêtre de mots d'une taille donnée w de part et d'autre du mot w_t à la position courante t. On obtient l'équivalent dans les réseaux neuronaux en concaténant les plongements des mots que l'on veut utiliser comme contexte. Cette concaténation est notée par :

$$I_t = [E(v(w_{t-w}))...E(v(w_t))...E(v(w_{t+w}))].$$

En utilisant ce contexte en entrée, avec la même notation utilisée dans (Jordan, 1989; Mesnil *et al.*, 2013), la sortie de la couche cachée d'un réseau de Jordan devient :

$$h_t = \Sigma(I_t \cdot H + y_{t-1} \cdot R)$$

Le calcul de la sortie reste inchangé, et on peut modifier de façon analogue un réseau d'Elman.

L'apprentissage des RNN décrits consiste dans l'estimation des paramètres $\Theta = (E, H, O, R)$, plus les biais que nous avons omis pour rendre les formules plus lisibles. Pour cela nous utilisons l'algorithme de propagation en arrière avec *vitesse* (Bengio, 2012), plus certaines recommandations proposées également par (Bengio, 2012).

2. On note que dans leur formulation originelle dans (Elman, 1990; Jordan, 1989), les RNN n'avaient pas une couche de plongements. Ces réseaux étaient en effet appliqués à des données qui se représentent naturellement avec des vecteurs à valeurs continues (e.g. le signal audio). Pour des applications au TAL, l'introduction d'une couche de plongements est nécessaire, précisément pour transformer des données à valeurs discrètes en valeurs continues.

3. Étant donnée l'entrée x, la sigmoïde est définie par $f(x) = \frac{1}{1+e^{-x}}$

4. Pour un ensemble de valeurs numériques $l \in L$, le *Softmax* de cet ensemble vaut $g(l) = \frac{e^l}{\sum_{l \in L} e^l}$

2.2 Variantes d'apprentissage

Un choix très important pour l'apprentissage des RNN concerne l'algorithme de propagation arrière. En effet, étant donnée leur architecture récurrente, pour apprendre proprement un RNN il faudrait utiliser l'algorithme dit de propagation arrière à travers le temps (*Back-propagation through time*) (Werbos, 1990), censé permettre d'apprendre des contextes arbitrairement longs. Cependant, (Mikolov *et al.*, 2011) a montré que les RNN pour les modèles des langues apprennent mieux avec seulement 5 étapes passées. Ceci pourrait être dû au fait que, dans les tâches de TAL, l'information passée retenue en mémoire par un RNN grâce à sa connexion récurrente se disperse après quelques étapes seulement. Aussi, du moins dans le cas des RNN de Jordan, garder en mémoire un contexte d'étiquettes prédites arbitrairement long ne donne pas en général une garantie d'amélioration des performances, un contexte plus long étant aussi plus bruité.

Puisque l'algorithme de propagation arrière à travers le temps est plus coûteux que l'algorithme classique, (Mesnil *et al.*, 2013) a préféré utiliser explicitement l'information contextuelle donnée à la connexion récurrente dans les RNN, et utiliser l'algorithme de propagation arrière classique, sans perte de performance. Dans cet article, nous adoptons la même stratégie d'apprentissage. Quand on utilise explicitement les étiquettes prédites aux étapes précédentes dans un réseau de Jordan, la sortie de la couche cachée est calculée par :

$$h_t = \Sigma(I_t \cdot H + [y_{t-c+1} y_{t-c+2} ... y_{t-1}] \cdot R)$$

où c est la taille du contexte "côté sortie" utilisé explicitement dans le réseau. Une modification équivalente permet de prendre en compte les couches cachées précédentes dans un RNN d'Elman.

2.3 Nouvelles variantes de RNN

Comme nous l'avons déjà expliqué en introduction, la variante principale de RNN que nous proposons comporte une connexion récurrente entre la couche de sortie et la couche d'entrée. Cette modification simple dans l'architecture des RNN a des conséquences importantes sur le modèle et, comme nous le verrons, sur les résultats. Ces conséquences ont été mentionnées dans l'introduction. Nous reprenons ici plus en détails les plus importantes d'entre elles.

La conséquence la plus importante de notre architecture est sa capacité d'utiliser des plongements pour les étiquettes, de la même façon que pour les mots. En effet, la première couche des réseaux est une table de "look-up" qui sert à transformer les représentations creuses "one-hot" en représentations distributionnelles. Ces dernières peuvent encoder des traits et des propriétés syntaxiques et sémantiques très attractives, comme cela a été démontré avec *word2vec* (Mikolov *et al.*, 2013a). Ces traits et propriétés peuvent être appris à partir de données étiquetées de la même façon que pour les mots. Dans cet article, on se contentera d'utiliser les séquences d'étiquettes associées aux séquences de mots. Mais cette approche pourrait aussi être étendue à des cas plus sophistiqués, en exploitant des informations structurées quand elles sont disponibles. On pourrait ansi exploiter de la même façon des arbres syntaxiques, des entités nommées étendues ou des relations entre entités nommées. L'idée d'utiliser des plongements pour les étiquettes n'est pas nouvelle en soi : (Chen & Manning, 2014) l'a introduite accessoirement dans le contexte de l'analyse syntaxique en dépendances, ce qui a donné un analyseur très efficace. L'originalité de notre travail, qui va au delà de cette première approche, est de pré-apprendre les plongements des étiquettes comme on le fait pour les mots. Aussi, nous utilisons plusieurs étiquettes comme contexte, ce qui va permettre d'apprendre d'une façon plus fine les dépendances entre étiquettes.

En utilisant le même formalisme que précédemment, nous notons E_w la matrice des plongements des mots, et E_l celle des plongements des étiquettes. Nous notons alors :

$$I_t = [E_w(v(w_{t-w}))...E_w(v(w_t))...E_w(v(w_{t+w}))]$$

la concaténation des vecteurs qui représentent le contexte des mots en entrée à la position t d'une séquence, alors que :

$$L_t = [E_l(v(y_{t-c+1}))E_l(v(y_{t-c+2}))...E_l(v(y_{t-1}))]$$

est la concaténation des vecteurs qui représentent les étiquettes prédites aux c positions précédentes. La sortie de la couche cachée est alors calculée comme suit :

$$h_t = \Sigma([I_t L_t] \cdot H)$$

À nouveau, $[\cdot]$ indique la concaténation de matrices et nous avons omis les biais pour rendre les formules plus simples. Les autres couches sont calculées comme précédemment. Comme nous pouvons le noter, une différence importante par rapport au réseau de Jordan est qu'ici les informations provenant des mots et des étiquettes ne sont pas *mélangées* en les sommant. La concaténation implique en effet que les deux informations sont données en entrées à la couche cachée tout en restant séparées [5]. Puisque nous utilisons le même nombre de neurones dans la couche cachée de tous les réseaux pour que les résultats soient comparables, dans notre variante de RNN chaque neurone de la couche cachée reçoit en entrée plus d'information que dans un réseau d'Elman ou de Jordan. Nous faisons l'hypothèse que cette architecture modélise de façon plus efficace les dépendances entre étiquettes grâce aux capacités des réseaux neuronaux d'apprendre des traits "internes" au niveau de la couche cachée.

Une autre conséquence importante de la modification introduite par notre variante de RNN est une robustesse accrue du modèle par rapport aux erreurs de prédictions. C'est une conséquence directe de l'utilisation des plongements d'étiquettes. Puisque nous utilisons plusieurs étiquettes prédites comme contexte (voir L_t plus haut), au moins lors des dernières étapes de l'apprentissage (quand le modèle est suffisamment précis), il est assez improbable que plusieurs erreurs de prédiction subsistent dans le même contexte. Même dans ce cas, grâce aux propriétés des plongements (Mikolov *et al.*, 2013b), les étiquettes erronées auront des représentations assez proches de celles des étiquettes attendues. En reprenant un exemple cité dans (Mikolov *et al.*, 2013b), si on utilise *Paris* à la place de *Rome*, cela n'a aucun effet pour plusieurs tâches de TAL : ce sont tous les deux des noms propres pour un étiquetage POS, des noms de lieu pour une extraction d'entités nommées, etc. L'utilisation de plongements d'étiquettes permet d'avoir la même robustesse sur les étiquettes, et donc dans le contexte.

Il ne se passe en général pas la même chose avec un modèle de Jordan, parce que la connexion récurrente entre la couche de sortie et la couche cachée dans ce type de RNN oblige à utiliser soit la distribution de probabilités sur les étiquettes, soit la représentation creuse "one-hot" des données en sortie par le Softmax [6]. Avec la représentation creuse, il est clair qu'une erreur affecte plus la sortie du réseau, puisque la seule valeur non nulle est dans ce cas mal positionnée. Utiliser la distribution de probabilités sur les étiquettes pourrait donner l'impression d'une certaine souplesse dans la fonction de décision. Cependant nous avons constaté expérimentalement que la masse des probabilités est concentrée dans une ou très peu d'étiquettes seulement. La distribution ne donne donc en fait pas beaucoup plus d'information que la représentation creuse. Dans tous les cas, les plongements d'étiquettes garantissent une souplesse bien plus attractive.

Il est important de noter cependant un autre point de vue sur la façon dont un réseau de Jordan calcule la sortie de la couche cachée. Comme nous l'avons décrit plus haut, celle-ci est calculée comme

$$h_t = \Sigma(I_t \cdot H + [y_{t-c+1}y_{t-c+2}...y_{t-1}] \cdot R)$$

5. Dans les réseaux de Jordan (Jordan, 1989), cela n'était pas nécessaire puisque les données en entrée et en sortie étaient de même nature (signal audio). En TAL, mais plus particulièrement dans les tâches d'étiquetage de séquences comme celles étudiées dans cet article, entrées et sorties sont différentes, ainsi il n'est pas justifié de les mélanger comme c'est le cas dans le réseau de Jordan.

6. En fait, rien n'empêche de transformer une telle sortie en un index que l'on peut utiliser pour sélectionner le plongement de l'étiquette correspondante. Cela équivaudrait à notre variante de RNN mais n'a pas encore été fait à notre connaissance.

Puisque chaque $y_i, i = t-1, \ldots, t-c+1$ est un vecteur creux, indépendamment du fait qu'il soit un vecteur "one-hot" ou une distribution de probabilités biaisée, nous pouvons interpréter la multiplication de chaque y_i par la matrice R comme la sélection d'un plongement pour l'étiquette correspondante, comme nous le faisons dans notre variante de RNN. Même avec cette interprétation, il y a une différence importante entre le réseau de Jordan et notre variante. En effet, comme on peut le voir dans la formule, une fois que la sélection du plongement pour l'étiquette a été faite avec $y_i \cdot R$, le résultat ne participe pas directement à la transformation linéaire opérée par la matrice H. Seul le contexte au niveau de mots est multiplié par H ($I_t \cdot H$). Le résultat de cette multiplication est additionné au résultat de $y_i \cdot R$. Dans un certain sens donc, le réseau de Jordan modélise l'interaction entre mots et étiquettes avec cette somme. Ensuite la fonction non-linéaire de la couche cachée est calculée (Σ). Nous estimons que, dans un réseau de Jordan (Jordan, 1989), cela avait du sens surtout parce que l'information en entrée et l'information en sortie étaient de même nature. Mais dans des tâches d'étiquetage de séquences, les mots et les étiquettes sont de nature différente, une telle modélisation n'est donc plus aussi bien fondée.

Dans notre variante, en revanche, on sélectionne d'abord le plongement de chaque étiquette avec $E[v(y_i)]$ [7]. Ensuite on concatène les plongements des mots I_t avec les plongements des étiquettes L_t. C'est seulement après que nous appliquons la transformation linéaire en multipliant par la matrice H. Puis, la fonction non-linéaire est appliquée, comme c'était le cas dans le réseau de Jordan. Dans notre variante donc, les étiquettes subissent toujours deux transformations : i) la conversion de "one-hot" en plongements, ii) la transformation linéaire en multipliant les plongements par la matrice H. De plus, nous laissons la couche cachée apprendre les interactions entre les mots et les étiquettes.

La seconde variante de RNN que nous proposons combine les caractéristiques d'un réseau d'Elman et de notre première variante. Cela permet de garder en mémoire l'état interne du réseau aux étapes de traitements précédentes et d'exploiter en même temps tous les avantages décrits pour l'utilisation d'un contexte d'étiquettes prédites, fourni sous forme de représentations distributionnelles. Dans cette seconde variante, la seule différence réside dans le calcul de la sortie de la couche cachée, pour laquelle nous utilisons la concaténation des c sorties précédentes de cette même couche cachée :

$$h_t = \Sigma([I_t L_t] \cdot H + [h_{t-c+1} h_{t-c+2} ... h_{t-1}] \cdot R)$$

Le reste du réseau fonctionne exactement comme dans la première variante.

2.4 Réseaux *forward*, *backward* et bidirectionnels

Nous avons étudié le comportement de tous les réseaux décrits jusqu'ici en trois versions : *forward*, *backward* et bidirectionnel (Schuster & Paliwal, 1997). La version *forward* est celle que nous venons de détailler. La version *backward* est équivalente, à la seule différence près qu'elle traite les séquences dans le sens inverse par rapport à la version *forward*, c'est-à-dire de la fin vers le début. La version *backward* permet donc de prédire l'étiquette à la position t d'une séquence, étant donnés les mêmes mots que la version *forward* et les c étiquettes futures (dans le réseau de Jordan et notre première variante), ou les c états futurs de la couche cachée (dans le réseau d'Elman), ou les c étiquettes futures associées aux c états futurs de la couche cachée (dans notre seconde variante).

La version bidirectionnelle utilise en même temps l'information passée et future mémorisées par les deux réseaux *forward* et *backward*, quel que soit le type de réseau. Le fonctionnement de cette troisième version est donc forcément différent de celui des autres : i) d'abord le réseau *backward* est utilisé pour prédire les étiquettes et/ou les états futur(e)s de la couche cachée, selon le type de réseau. ii) les étiquettes et/ou les états de la couche cachée sont initialisés avec des valeurs "fillers" pour la phase *forward*. iii) un modèle global parcourt la séquence en avant en utilisant les étiquettes et/ou les états passés et futurs de la couche cachée. Cette procédure est décrite dans l'article de référence sur les réseaux neuronaux bidirectionnels (Schuster & Paliwal, 1997).

7. Dans ce cas, le vecteur y_i est explicitement converti en une représentation "one-hot".

2.5 Complexité des RNN

Avant de passer à l'évaluation de nos modèles, nous fournissons une analyse de leur complexité en termes de nombre de paramètres à trouver. Dans un modèle de Jordan nous en avons :

$$\{|V| \times D\}_E + \{((2w+1)D + c|O|) \times |H|\}_H + \{|H| \times |O|\}_O$$

Nous avons séparé les paramètres de chaque couche par $\{\}_E$, $\{\}_H$ et $\{\}_O$, pour la couche des plongements, la couche cachée et la couche de sortie, respectivement. $|V|$ est la taille du dictionnaire de mots, D la taille des plongements, $|H|$ et $|O|$ sont les tailles de la couche cachée et de sortie, respectivement, w est la taille (ou le rayon) de la fenêtre de mots utilisée comme contexte en entrée du réseau [8], c est la taille du contexte d'étiquettes, multipliée donc par la taille du dictionnaire des étiquettes qui correspond à celle de la couche de sortie $|O|$.

De la même façon, pour un réseau d'Elman et pour notre première variante il y en a respectivement :

$$\{|V| \times D\}_E + \{((2w+1)D + c|H|) \times |H|\}_H + \{|H| \times |O|\}_O$$

$$\{|V| \times D\}_E + \{|O| \times D + ((2w+1)D + cD) \times |H|\}_H + \{|H| \times |O|\}_O$$

Comme nous pouvons le voir, la seule différence entre les nombres de paramètres du RNN de Jordan et de celui d'Elman reside dans les facteurs $c|O|$ et $c|H|$. Leur différence en complexité dépend donc de la taille de la couche de sortie par rapport à la taille de la couche cachée. Puisque, dans les tâches d'étiquetage, cette dernière est en général plus grande, le réseau d'Elman est en général plus lourd.

La différence entre les paramètres du RNN de Jordan et de notre première variante tient dans les facteurs $|O| \times D$ et cD. Les premiers sont dûs à l'utilisation des plongements d'étiquettes [9], le second à l'utilisation des plongements d'étiquettes comme entrées de la couche cachée. Puisque souvent D et O ont des valeurs du même ordre de grandeur, et grâce à l'utilisation d'operations optimisées sur les vecteurs et les matrices, nous n'avons pas remarqué de différence remarquable dans les temps d'apprentissage et d'étiquetage entre le réseau de Jordan et notre première variante.

Notre seconde variante, en revanche, a la complexité suivante :

$$\{|V| \times D\}_E + \{|O| \times D + ((2w+1)D + cD + c|H|) \times |H|\}_H + \{|H| \times |O|\}_O$$

Le terme additionel $c|H|$ par rapport à notre première variante est dû à la présence d'une connexion récurrente comme celle du réseau d'Elman. Malgré l'utilisation d'opérations optimisées sur les matrices et les vecteurs, cette variante est relativement plus lourde, avec un temps d'apprentissage et d'étiquetage plus important d'un facteur d'environ $1,5$ par rapport aux autres.

Les versions *forward* et *backward* des RNN ont exactement la même complexité. En revanche la version bidirectionnelle est plus complexe. En effet, pour pouvoir prendre en compte les informations contextuelles passées et futures, la taille de la couche cachée doit être doublée par rapport à la couche cachée du réseau simple. De plus, le modèle *backward* doit être utilisé aussi pour prédire l'information qui sera utilisée dans le modèle bidirectionnel comme information future. Sans recourir à une formule, nous avons observé un temps d'apprentissage et d'étiquetage plus important d'un facteur $1,5$ par rapport au modèle simple. Cependant, l'utilisation des deux modèles *forward* et *backward* pour initialiser les poids du modèle global fait que le modèle bidirectionnel n'a pas besoin du même nombre d'étapes pour converger, il est de ce fait appris en un temps équivalent à son pendant simple.

3 Évaluation

3.1 Corpus utilisés

Nous avons évalué nos modèles sur deux tâches :

8. ce contexte comprend donc w mots à gauche, w à droite, plus le mot à la position courante t
9. Nous utilisons des plongements de même taille D pour les mots et les étiquettes

	apprentissage		développement		test	
# phrases	12,908		1,259		3,005	
	mots	concepts	mots	concepts	mots	concepts
# mots	94,466	43,078	10,849	4,705	25,606	11,383
# vocab.	2,210	99	838	66	1,276	78
# OOV%	–	–	1.33	0.02	1.39	0.04

TABLE 1 – Statistiques sur le corpus MEDIA

Le corpus ATIS (*Air Travel Information System*) (Dahl *et al.*, 1994) a été collecté pour construire des systèmes de dialogues automatiques capables de donner des informations sur les vols aux États Unis. La représentation sémantique est basée sur la notion de "slot" d'un "frame" ; le but dans la tâche de compréhension de la parole (SLU) est d'associer chaque mot au "slot" correspondant. Un exemple de phrase de ce corpus est *"I want all the flights from Boston to Philadelphia today"* [10]. Les mots *Boston*, *Philadelphia* et *today* sont associés aux "slots" `DEPARTURE.CITY`, `ARRIVAL.CITY` et `DEPARTURE.DATE`, respectivement. Tous les autres mots de la phrase n'appartiennent à aucun "slot".

C'est une tâche assez simple qui date de 1993. Les données d'apprentissage sont constituées de 4978 phrases prises parmi celles sans dépendances de contexte dans les corpus `ATIS-2` et `ATIS-3`. Les données de test sont constituées de 893 phrases prises à l'intérieur des corpus `ATIS-3 NOV93` et `DEC94`. Il n'y a pas de données de développement officielles, nous avons donc isolé une partie des données d'apprentissage et les avons utilisées comme données de développement. Les dictionnaires de mots et d'étiquettes sont constitués de 1117 et 85 unités, respectivement. Nous renvoyons à (Dahl *et al.*, 1994) pour plus de détails.

Le corpus français MEDIA (Bonneau-Maynard *et al.*, 2006) a été créé pour l'évaluation de systèmes de dialogues automatiques destinés à fournir des informations touristiques sur les hôtels en France. Il est constitué de 1250 dialogues acquis avec le protocole *Wizard-of-OZ* où 250 orateurs ont suivi 5 scénarios de réservation différents. Les dialogues ont été transcrits et annotés manuellement suivant une ontologie de concepts riches. Des composants sémantiques peuvent être combinés pour former des étiquettes sémantiques complexes [11]. Outre la richesse de l'annotation utilisée, une autre source de difficultés pour l'étiquetage provient des phénomènes de co-référence. Certains mots ne peuvent pas être annotés correctement sans connaître le contexte du tour de parole précédent. Par exemple dans la phrase *"Oui, celui à moins de 50 euros par nuit"*, *celui* réfère à un hôtel introduit dans un tour de parole précédent. Les propriétés statistiques des données d'apprentissage, de développement et de test du corpus MEDIA sont données dans le tableau 1.

La tâche MEDIA peut être modélisée comme un étiquetage de séquences en utilisant la convention classique *BIO* (Ramshaw & Marcus, 1995). Dans la tâche ATIS chaque concept (ou "slot") correspond à un mot, il s'agit donc d'une tâche équivalente à un étiquetage en partie du discours (*POS tagging*).

3.2 Réglages et paramètres

Pour garantir une comparaison équitable entre tous les modèles, nous avons utilisé les mêmes réglages que dans (Yao *et al.*, 2013; Mesnil *et al.*, 2013; Vukotic *et al.*, 2015) pour les tâches ATIS et MEDIA. Les plongements ont donc une taille de 200, de 100 pour la couche cachée. Nous avons également utilisé la même taille de fenêtre de mots $w = 3$ et $c = 6$ pour le contexte d'étiquettes prédites dans les RNN de Jordan et nos deux variantes. Nous avons partout utilisé la même "tokenisation", qui

10. Je voudrais tous les vols d'aujourd'hui de Boston à Philadelphie

11. Par exemple l'étiquette `localisation` peut être combinée avec les composants `ville`, `distance-relative`, `localisation-relative-générale`, `rue` etc.

Modèle	Mesure F1		
	forward	*backward*	bidirectionnel
Mots			
(Mesnil *et al.*, 2013) E-RNN	93.65%	92.12%	–
(Mesnil *et al.*, 2013) J-RNN	93.77%	93.31%	93.98%
(Mesnil *et al.*, 2015) E-RNN	**94.98%**	–	94.73%
E-RNN	93.41%	93.18%	93.41%
J-RNN	93.61%	93.55%	93.74%
I-RNN	**93.84%**	93.56%	93.73%
I+E-RNN	93.74%	93.44%	93.60%
Classes			
(Vukotic *et al.*, 2015) E-RNN	96.16%	–	–
(Vukotic *et al.*, 2015) CRF	–	–	95.23%
(Yao *et al.*, 2013) E-RNN	96.04%	–	–
(Mesnil *et al.*, 2015) E-RNN	96.24%	–	**96.29%**
(Mesnil *et al.*, 2015) CRF	–	–	95.16%
E-RNN	94.73%	93.61%	94.71%
J-RNN	94.94%	94.80%	94.89%
I-RNN	**95.21%**	94.64%	94.75%
I+E-RNN	94.84%	94.61%	94.79%

TABLE 2 – Résultats sur le corpus ATIS

Modèle	Mesure F1		
	forward	*backward*	bidirectionnel
(Vukotic *et al.*, 2015) E-RNN	81.94%	–	–
(Vukotic *et al.*, 2015) J-RNN	83.25%	–	–
(Vukotic *et al.*, 2015) CRF	–	–	86.00%
E-RNN	82.64%	82.61%	83.13%
J-RNN	83.06%	83.74%	84.29%
I-RNN	**84.91%**	**86.28%**	**86.71%**
I+E-RNN	84.58%	85.84%	86.21%

TABLE 3 – Résultats sur le corpus MEDIA

consiste essentiellement en une conversion des caractères en minuscule. Nous avons fait des choix plus "classiques" pour la fonction d'activation de la couche cachée et la régularisation que dans les travaux les plus récents : nous utilisons la sigmoïde et la régularisation $L2$ respectivement, alors que (Yao *et al.*, 2013; Mesnil *et al.*, 2013) et (Vukotic *et al.*, 2015) utilisent la fonction linéaire rectifiée et la régularisation *dropout* [12] (Bengio, 2012; Srivastava *et al.*, 2014). Nous avons entraîné tous les RNN avec la procédure suivante :

i) Nous avons d'abord entraîné deux modèles de langage neuronal (NNLM) pour obtenir les plongements des mots et des étiquettes. Ces NNLM ont la même architecture que celui de (Bengio *et al.*, 2003), à la seule différence près que nous utilisons une fenêtre de mots autour de la position courante comme contexte, au lieu de prendre seulement les mots dans le passé. Pour les étiquettes, nous utilisons seulement les étiquettes dans le passé comme contexte.

ii) Nous entraînons ensuite les différents RNN en utilisant pour tous les mêmes plongements.

Nous entraînons le modèle pour les plongements des mots pendant 20 unités de temps, celui pour les plongements des étiquettes pendant 10, les modèles *forward* et *backward* pendant 20 et les modèles bidirectionnel pendant 8. Ces valeurs ont été optimisées sur les données de développement.

3.3 Résultats

L'évaluation de tous les modèles évoqués dans cet article est donnée dans les tableaux 2 et 3 en termes de F1-mesure. Dans tous les tableaux, nos versions des RNN d'Elman et Jordan sont indiquées avec

12. Nous avons testé aussi la fonction linéaire rectifiée et la régularisation *dropout*. Cela donne des résultats comparables ou meilleurs au prix d'une couche cachée bien plus grande. De plus l'apprentissage n'est pas stable avec toutes les valeurs des certains paramètres (taux d'apprentissage notamment). Pour ces raisons, pour les expériences présentées dans cet article nous avons préféré la sigmoïde et la régularisation $L2$.

E−RNN et J−RNN. Nos deux variantes sont désignées par I−RNN et I+E−RNN. Le tableau 2 montre les résultats obtenus sur le corpus ATIS. Dans la partie haute du tableau sont montrés les résultats obtenus en n'utilisant que les mots comme entrées des modèles. Dans la partie basse (Classes), les classes des mots disponibles pour cette tâche sont ajoutées aux données. Ces classes identifient les noms de villes, d'aéroports et les expressions de temps (dates et horaires de départ et d'arrivée) ; elles permettent aux modèles de faire abstraction des mots spécifiques qui déclenchent certains concepts [13].

Notons avant toute analyse que certains de nos résultats sur ATIS ne sont pas parfaitement comparables avec ceux publiés dans la littérature pour les raisons suivantes : *i)* les modèles à l'état-de-l'art utilisent une fonction d'activation de type *Rectified Linear function* [14] et la régularisation *dropout*. Nos modèles utilisent la sigmoïde et la regularisation $L2$. *ii)* Pour les expériences sur le corpus ATIS, nous avons utilisé environ 18% des données d'apprentissage, choisies au hasard, comme données de développement ; nous disposions donc d'un peu moins de données d'apprentissage [15]. *iii)* Les travaux publiés avec lesquels nous nous comparons ne décrivent pas toujours en détails comment les classes disponibles pour cette tâche sont intégrées dans leur modèle. *iv)* Les réglages de nos modèles ne correspondent pas toujours avec ceux des modèles publiés. Malgré tout, nos expériences sont fiables et probantes : *i)* d'une part, certains de nos résultats, obtenus dans des conditions légèrement handicapantes, sont aussi bons que l'état-de-l'art, voire meilleurs dans certains cas ; *ii)* d'autre part, nous fournissons une comparaison équitable avec nos propres versions de tous les RNN évoqués, ce qui nous permet de nous comparer à l'état-de-l'art et non pas à des modèles "baseline".

Les résultats de la partie haute (*Mots*) du tableau 2 montrent que le meilleur modèle sur la tâche ATIS est (Mesnil *et al.*, 2015). Notons au passage que la façon dont les auteurs ont amélioré les performances de (Mesnil *et al.*, 2013) (dû en partie aux mêmes auteurs) n'est pas très claire. Dans (Mesnil *et al.*, 2013) les auteurs obtiennent leurs meilleurs résultats avec un réseau de Jordan, alors que dans (Mesnil *et al.*, 2015) c'est un réseau d'Elman qui atteint les meilleures performances. Avec nos expériences réalisées dans les mêmes conditions, nous avons pu constater que le gain montré dans (Mesnil *et al.*, 2015) n'est pas imputable à la seule optimisation des paramètres. Nous constatons que nos modèles d'Elman et de Jordan sont à peu près équivalents à ceux de (Mesnil *et al.*, 2013). Notre variante principale de RNN I−RNN obtient le deuxième meilleur résultat absolu (93.84 en gras). Notre deuxième variante est à peu près équivalente à un réseau de Jordan sur cette tâche.

Les résultats de la partie basse (*Classes*) du tableau 2, obtenus en utilisant les mots et les classes en entrée des RNN, remarquablement meilleurs que ceux obtenus avec les mots seuls, suivent à peu près le même comportement. Nous attribuons les différences entre nos résultats et ceux de l'état-de-l'art aux mêmes raisons mentionnées plus haut. A titre de comparaison, nous montrons aussi dans cette partie du tableau des résultats obtenus avec des CRF. De façon générale, sur la tâche ATIS, quelles que soient les données d'entrée, nous pouvons remarquer que tous les résultats sont assez bons. Cette tâche est de fait relativement simple, les dépendances entre étiquettes peuvent être facilement résolues à cause de la bonne correspondance entre les concepts et les mots (un concept correspond à un mot). Ces conditions ne permettent pas à nos variantes des RNN d'exprimer leur potentiel lié à la modélisation des étiquettes et des dépendances entre étiquettes sous forme de plongements. Cette limitation est confirmée par les résultats obtenus par (Vukotic *et al.*, 2015) et (Mesnil *et al.*, 2015) avec les CRF. En effet, malgré la richesse d'information utilisée dans la fonction de décision des RNN pour prédire la prochaine étiquette, elle reste locale -au sens où elle ne prend pas en compte l'étiquetage de la séquence entière comme critère de décision. En revanche, les CRF prennent en compte les dépendances entre étiquettes dans la séquence (Lafferty *et al.*, 2001). Le fait qu'ils soient moins efficaces, même que les RNN *forward* simples, sur cette tâche, est un indice clair du manque de dépendances entre étiquettes. Nous remarquons par ailleurs que c'est toujours la première de nos

13. Par exemple les villes *Boston* et *Philadelphie* dans l'exemple plus haut sont converties en CITY−NAME. Même si le modèle n'a jamais vu le mot *Boston* pendant l'apprentissage, s'il a vu au moins un nom de ville, il sera capable de bien annoter *Boston* comme une *ville-de-départ* grâce au contexte discriminant donné par la préposition *from*.

14. f(x) = max(0,x)

15. Cela pour éviter le coût computationnel d'une validation croisée.

deux variantes I−RNN qui obtient le meilleurs résultats parmi nos RNN. Cela montre l'intérêt de l'utilisation des plongements des étiquettes, même sur une tâche relativement simple.

Dans le tableau 3 nous montrons les résultats atteints sur la tâche MEDIA. Cette tâche est plus difficile, à cause de la richesse de l'annotation sémantique, mais aussi des co-références et des concepts qui tiennent sur plusieurs mots. Cette propriété crée des dépendances entre étiquettes relativement longues, qui ne peuvent pas être prises en compte par des modèles simples. La difficulté de la tâche est illustrée par ailleurs par le niveau des performances répertoriées dans le tableau 3 (environ 10 points de F1-mesure en moins par rapport à ATIS).

Comme nous pouvons le constater dans ce tableau, sur cette tâche les RNN simples sont bien moins efficaces que les CRF, ce qui pouvait être attendu à cause de leur fonction de décision locale. Nous remarquons également que nos versions des RNN d'Elman et de Jordan sont équivalentes, voire meilleures dans le cas du RNN d'Elman, aux RNN de (Vukotic *et al.*, 2015), ces derniers étant état-de-l'art. Plus important, les résultats sur MEDIA montrent que, dans ce contexte où une modélisation des dépendances entre étiquettes est cruciale, nos variantes de RNN marchent remarquablement mieux que les réseaux classiques d'Elman et de Jordan, et ce que ce soit dans leur version *forward*, *backward* ou bidirectionnelle. Dans certains cas, notamment celui des RNN bidirectionnels, ils donnent même de meilleurs résultats que les CRF. Nous interprétons cela comme une capacité de nos RNN à pallier une fonction de décision locale par une meilleure modélisation des dépendances entre étiquettes.

Nous notons aussi que nous avons mené des expériences sans pre-apprendre les plongements des étiquettes. Dans ces expériences, les plongements sont appris en même temps que la tâche d'étiquetage de séquences. Par manque de place nous ne montrons pas ces résultats, mais ils ne sont pas significativement différents de ceux obtenus avec le pre-apprentissage des plongements des étiquettes. Cela montre une fois de plus l'efficacité de l'utilisation des plongements d'étiquettes. De plus, cela donne une application plus ample à l'utilisation des plongements côté sortie en général. Des corpus d'étiquettes ne sont certes pas aussi disponibles que le sont les corpus des mots. Mais, comme nos variantes sont efficaces même sans le pre-apprentissage des plongements d'étiquettes, un tel manque n'est pas pénalisant pour nos modèles neuronaux.

Il apparaît au vu de ces résultats que des RNN utilisant des plongements pour les étiquettes peuvent modéliser de façon efficace les dépendances entre étiquettes, même quand ces dépendances sont relativement simples (comme pour ATIS). Nos variantes de RNN sont pour cette raison plus efficaces que les RNN classiques d'Elman et de Jordan. Nous avons par ailleurs effectué des expériences du même type sur la tâche d'étiquetage en parties du discours du corpus FTB (Abeillé *et al.*, 2003) qui confirment ces conclusions. Pour cette tâche, la hiérarchie établie ici entre les RNN est respectée, nos variantes de réseaux se montrant meilleurs que ceux d'Elman et de Jordan.

4 Conclusions et Perspectives

Dans cet article nous avons étudié plusieurs types de réseaux neuronaux récurrents (RNN) pour des tâches d'étiquetage de séquences. Nous avons proposé deux nouvelles variantes de RNN pour mieux modéliser les dépendances entre étiquettes et nous les avons comparées aux variantes existantes de type Jordan et Elman. Nous avons expliqué en détails pourquoi et comment nos nouvelles variantes donnent des avantages par rapport aux autres RNN. Nous avons évalué tous les RNN évoqués de façon équitable, sur deux tâches différentes de compréhension de la parole. Les résultats montrent que nos nouvelles variantes de RNN sont plus efficaces que les versions initiales.

Références

ABEILLÉ A., CLÉMENT L. & TOUSSENEL F. (2003). Building a Treebank for French. In

Treebanks : Building and Using Parsed Corpora, p. 165–188. Springer.

F. BENARMARA, N. HATOUT, P. MULLER & S. OZDOWSKA, Eds. (2007). *Actes de TALN 2007 (Traitement automatique des langues naturelles)*, Toulouse. ATALA, IRIT.

BENGIO Y. (2012). Practical recommendations for gradient-based training of deep architectures. *CoRR*, **abs/1206.5533**.

BENGIO Y., DUCHARME R., VINCENT P. & JAUVIN C. (2003). A neural probabilistic language model. *JOURNAL OF MACHINE LEARNING RESEARCH*, **3**, 1137–1155.

BONNEAU-MAYNARD H., AYACHE C., BECHET F., DENIS A., KUHN A., LEFÈVRE F., MOSTEFA D., QUGNARD M., ROSSET S. & SERVAN, S. VILANEAU J. (2006). Results of the french evalda-media evaluation campaign for literal understanding. In *LREC*, p. 2054–2059, Genoa, Italy.

CHEN D. & MANNING C. (2014). A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, p. 740–750, Doha, Qatar : Association for Computational Linguistics.

COLLOBERT R. & WESTON J. (2008). A unified architecture for natural language processing : Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, p. 160–167, New York, NY, USA : ACM.

COLLOBERT R., WESTON J., BOTTOU L., KARLEN M., KAVUKCUOGLU K. & KUKSA P. (2011). Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, **12**, 2493–2537.

DAHL D. A., BATES M., BROWN M., FISHER W., HUNICKE-SMITH K., PALLETT D., PAO C., RUDNICKY A. & SHRIBERG E. (1994). Expanding the scope of the atis task : The atis-3 corpus. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, p. 43–48, Stroudsburg, PA, USA : Association for Computational Linguistics.

DE MORI R., BECHET F., HAKKANI-TUR D., MCTEAR M., RICCARDI G. & TUR G. (2008). Spoken language understanding : A survey. *IEEE Signal Processing Magazine*, **25**, 50–58.

G. DIAS, Ed. (2015). *Actes de TALN 2015 (Traitement automatique des langues naturelles)*, Caen. ATALA, HULTECH.

ELMAN J. L. (1990). Finding structure in time. *COGNITIVE SCIENCE*, **14**(2), 179–211.

JORDAN M. I. (1989). Serial order : A parallel, distributed processing approach. In J. L. ELMAN & D. E. RUMELHART, Eds., *Advances in Connectionist Theory : Speech*. Hillsdale, NJ : Erlbaum.

LAFFERTY J., MCCALLUM A. & PEREIRA F. (2001). Conditional random fields : Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML)*, p. 282–289, Williamstown, MA, USA.

LAIGNELET M. & RIOULT F. (2009). Repérer automatiquement les segments obsolescents à l'aide d'indices sémantiques et discursifs. In *Actes de TALN 2009 (Traitement automatique des langues naturelles)*, Senlis : ATALA LIPN.

LANGLAIS P. & PATRY A. (2007). Enrichissement d'un lexique bilingue par analogie. In (Benarmara *et al.*, 2007), p. 101–110.

MESNIL G., DAUPHIN Y., YAO K., BENGIO Y., DENG L., HAKKANI-TUR D., HE X., HECK L., TUR G., YU D. & ZWEIG G. (2015). Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*.

MESNIL G., HE X., DENG L. & BENGIO Y. (2013). Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *Interspeech 2013*.

MIKOLOV T., CHEN K., CORRADO G. & DEAN J. (2013a). Efficient estimation of word representations in vector space. *CoRR*, **abs/1301.3781**.

MIKOLOV T., KARAFIÁT M., BURGET L., CERNOCKÝ J. & KHUDANPUR S. (2010). Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, p. 1045–1048.

MIKOLOV T., KOMBRINK S., BURGET L., CERNOCK˘ J. & KHUDANPUR S. (2011). Extensions of recurrent neural network language model. In *ICASSP*, p. 5528–5531 : IEEE.

MIKOLOV T., YIH W. & ZWEIG G. (2013b). Linguistic regularities in continuous space word representations. In *Human Language Technologies : Conference of the North American Chapter of the Association of Computational Linguistics*, p. 746–751.

RAMSHAW L. & MARCUS M. (1995). Text chunking using transformation-based learning. In *Proceedings of the 3rd Workshop on Very Large Corpora*, p. 84–94, Cambridge, MA, USA.

SCHUSTER M. & PALIWAL K. (1997). Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, **45**(11), 2673–2681.

SERETAN V. & WEHRLI E. (2007). Collocation translation based on sentence alignment and parsing. In (Benarmara *et al.*, 2007), p. 401–410.

SRIVASTAVA N., HINTON G., KRIZHEVSKY A., SUTSKEVER I. & SALAKHUTDINOV R. (2014). Dropout : A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, **15**, 1929–1958.

VUKOTIC V., RAYMOND C. & GRAVIER G. (2015). Is it time to switch to word embedding and recurrent neural networks for spoken language understanding ? In *InterSpeech*, Dresde, Germany.

WERBOS P. (1990). Backpropagation through time : what does it do and how to do it. In *Proceedings of IEEE*, volume 78, p. 1550–1560.

YAO K., ZWEIG G., HWANG M.-Y., SHI Y. & YU D. (2013). : Interspeech.