Modélisation de dépendances entre étiquettes dans les réseaux neuronaux récurrents

Marco Dinarelli, Yoann Dupont

▶ **To cite this version:**

Marco Dinarelli, Yoann Dupont. Modélisation de dépendances entre étiquettes dans les réseaux neuronaux récurrents. Revue TAL : traitement automatique des langues, 2017, 58 (1). hal-01579114

HAL Id: hal-01579114
https://hal.science/hal-01579114

Submitted on 30 Aug 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Modélisation de dépendances entre étiquettes dans les réseaux neuronaux récurrents

Marco Dinarelli[*]**, Yoann Dupont**[*] [**]

[*] *Lattice (UMR 8094), CNRS, ENS Paris, Université Sorbonne Nouvelle - Paris 3*
PSL Research University, USPC (Université Sorbonne Paris Cité)
1 rue Maurice Arnoux, 92120 Montrouge, France
[**] *Expert System France, 207 rue de Bercy, 75012 Paris*
marco.dinarelli@ens.fr, yoa.dupont@gmail.com

RÉSUMÉ. Les réseaux neuronaux récurrents (RNN) se sont montrés très efficaces dans plusieurs tâches de traitement automatique des langues. Cependant, leur capacité à modéliser *l'étiquetage de séquences* reste limitée. Cette limitation a conduit la recherche vers la combinaison des RNN avec des modèles déjà utilisés avec succès dans ce contexte, comme les CRF. Dans cet article, nous étudions une solution plus simple mais tout aussi efficace : une évolution du RNN de Jordan dans lequel les étiquettes prédites sont réinjectées comme entrées dans le réseau et converties en plongements, de la même façon que les mots. Nous comparons cette variante de RNN avec tous les autres modèles existants : Elman, Jordan, LSTM et GRU, sur deux tâches de compréhension de la parole. Les résultats montrent que la nouvelle variante, plus complexe que les modèles d'Elman et Jordan, mais bien moins que LSTM et GRU, est non seulement plus efficace qu'eux, mais qu'elle fait aussi jeu égal avec des modèles CRF plus sophistiqués.

ABSTRACT. Recurrent Neural Networks have proved effective on several NLP tasks. Despite such great success, their ability to model *sequence labeling* is still limited. This lead research toward solutions where RNNs are combined with models successfully employed in this context, like CRFs. In this work we propose a simpler solution: an evolution of the Jordan RNN, where labels are reinjected as input into the network and converted into embeddings, the same way as words. We compare this variant to all the other RNN models, Elman, Jordan, LSTM and GRU, on two tasks of Spoken Language Understanding. Results show that the new variant, which is more complex than Elman and Jordan RNNs, but far less than LSTM and GRU, is more effective than other RNNs and outperforms sophisticated CRF models.

MOTS-CLÉS : *RNN, modélisation de séquences, compréhension automatique de la parole.*

KEYWORDS: *RNNs, sequence modelling, spoken language understanding.*

1. Introduction

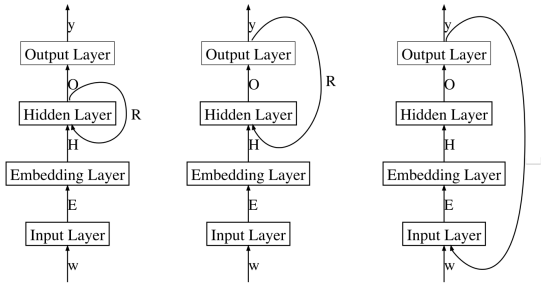
Dans les dernières années les réseaux neuronaux récurrents (RNN) (Jordan, 1989 ; Elman, 1990 ; Hochreiter et Schmidhuber, 1997) se sont montrés très efficaces dans plusieurs tâches de traitement automatique des langues (TAL) comme l'étiquetage des parties du discours, la segmentation, la reconnaissance des entités nommées, la compréhension automatique de la parole, la traduction automatique statistique et d'autres encore (Mikolov *et al.*, 2010 ; Mikolov *et al.*, 2011 ; Collobert et Weston, 2008 ; Collobert *et al.*, 2011 ; Yao *et al.*, 2013 ; Mesnil *et al.*, 2013 ; Vukotic *et al.*, 2015). L'efficacité de ces modèles vient de leur architecture récurrente, qui permet de garder une mémoire de l'information traitée dans le passé et de la réutiliser à l'étape courante.

Dans la littérature, plusieurs types d'architectures ont été proposés, notamment les RNN d'Elman (Elman, 1990) et ceux de Jordan (Jordan, 1989), aussi appelés RNN simples. 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 les positions précédentes d'une séquence.

Ces deux types de modèles sont limités pour l'encodage de contextes relativement longs (Bengio *et al.*, 1994). Pour cette raison les RNN de type *Long Short-Term Memory* (LSTM) ont été proposés (Hochreiter et Schmidhuber, 1997). Récemment, une variante simplifiée et plus efficace des LSTM a été introduite, utilisant des *Gated Recurrent Units*, d'où son nom : GRU (Cho *et al.*, 2014).

Malgré leur relative efficacité sur plusieurs tâches, les RNN présentent toujours des limitations dans leur capacité à modéliser les dépendances entre les unités de sortie, c'est-à-dire les étiquettes. Leurs fonctions de décision restent en effet intrinsèquement locales. Pour pallier cette limitation, des modèles hybrides *RNN+CRF* ont été testés (Huang *et al.*, 2015 ; Lample *et al.*, 2016 ; Ma et Hovy, 2016). Ces modèles atteignent des résultats à l'état de l'art, et bien que leur efficacité soit indéniable, il n'est pas clair qu'elle soit due aux modèles eux-mêmes ou aux conditions expérimentales particulièrement favorables dans lesquelles ils ont été évalués (notamment grâce à des plongements de mots entraînés sur de très grandes quantités de données avec des outils tels que *word2vec* (Mikolov *et al.*, 2013a) ou *GloVe* (Pennington *et al.*, 2014)).

L'intuition à la base de cet article est que les plongements permettent une modélisation plus efficace non seulement des mots, mais également des *étiquettes* et de leurs dépendances, qui sont cruciales dans certaines tâches d'étiquetage de séquences. Dans cet article, nous étudions une variante favorisant cette modélisation. Cette variante redonne 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. Puisque l'entrée du réseau est constituée d'indices qui sont utilisés pour sélectionner des plongements, dans notre variante les étiquettes sont converties en plongements de la même façon que les mots. Nous espérons, grâce à l'utilisation des plongements



(a) Elman (b) Jordan (c) Notre variante

Figure 1. *Schéma général des principaux RNN et de notre variante*

sur plusieurs étiquettes comme contexte, modéliser de façon efficace les dépendances entre étiquettes.

Un modèle similaire a été proposé par Dinarelli et Tellier (2016a) et Dinarelli et Tellier (2016c), et ensuite amélioré dans **?**. Nous décrivons ici une version améliorée de ce modèle suivant différentes dimensions : i) nous étudions l'effet de couches cachées plus sophistiquées telles que les *ReLU*, les *Leaky ReLU* et les *ReLU* paramétrés (He *et al.*, 2015) ; ii) nous utilisons la régularisation *dropout* (Srivastava *et al.*, 2014) dans les couches de plongements et les couches cachées ; iii) nous intégrons une couche de convolution sur les caractères pour modéliser des traits à un niveau plus fin que les mots ; iv) nous évaluons les modèles de façon exhaustive en montrant les effets des différents composants et niveaux d'information.

Idéalement, la variante de RNN proposée dans cette article peut être vue comme une évolution du modèle simple de Jordan, dans laquelle la connexion récurrente relie la couche de sortie à la couche d'entrée. Un schéma général des réseaux d'Elman, de Jordan et de la variante proposée est montré dans la figure 1, où w sont les mots, y les étiquettes, et E, H, O et R sont les paramètres du modèle. Tous les RNN mentionnés dans cet article sont étudiés dans leur version *en avant* (dits *forward* dans la suite de l'article), *en arrière* (*backward*) et bidirectionnelle (Schuster et Paliwal, 1997).

Nous évaluons ces 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). ATIS est relativement simple et ne nécessite pas une modélisation poussée des dépendances entre étiquettes. Cette tâche nous permet d'évaluer les modèles dans des conditions similaires à celles des tâches comme l'étiquetage des parties du discours, ou la reconnaissance des entités nommées telle que définie dans la *CoNLL Shared Task* 2003, toutes les deux largement utilisées comme références dans le TAL. MEDIA est une tâche plus difficile pour laquelle la capacité d'un modèle à prendre en compte les dépendances entre étiquettes est cruciale. Les résultats obtenus montrent que notre variante est aussi efficace que les meilleurs RNN actuels sur la tâche ATIS, tout en étant plus simple. Sur MEDIA notre variante obtient des résultats

bien meilleurs que tous les autres RNN, dépassant aussi les CRF, la référence pour l'étiquetage de séquences.

Le reste de l'article est organisé comme suit. Dans la section suivante, nous décrivons en détail les réseaux neuronaux récurrents, partant des modèles existants pour arriver à notre variante. La section 3 est consacrée à la présentation des corpus sur lesquels nous évaluons les modèles, aux paramétrages et choix architecturaux et aux résultats obtenus. La section 4 présente nos conclusions.

2. Réseaux neuronaux récurrents (RNN)

Dans cette section nous présentons l'état de l'art sur les RNN classiques, tels que les RNN d'Elman et de Jordan (Jordan, 1989 ; Elman, 1990), mais aussi certains plus sophistiqués comme les LSTM et les GRU (Hochreiter et Schmidhuber, 1997 ; Cho *et al.*, 2014), ainsi que les procédures d'apprentissage et d'inférence. Par souci de continuité dans l'article, nous présentons ici également notre nouvelle variante de RNN.

2.1. *RNN d'Elman et de Jordan*

Les RNN d'Elman et Jordan sont définis respectivement comme suit :

$$\mathbf{h_t}^{\text{Elman}} = \Phi(R\,\mathbf{h_{t-1}} + H\,\mathbf{I_t}) \quad [1]$$

$$\mathbf{h_t}^{\text{Jordan}} = \Phi(R\,\mathbf{y_{t-1}} + H\,\mathbf{I_t}) \quad [2]$$

La différence entre ces deux modèles tient au calcul des activités de la couche cachée, alors que, dans les deux cas, la couche de sortie est calculée avec :

$$\mathbf{y_t} = softmax(O\,\mathbf{h_t}) \quad [3]$$

où $\mathbf{h_t}$ et $\mathbf{y_t}$ sont respectivement la sortie de la couche cachée (que ce soit dans le modèle d'Elman, Jordan ou autres) et de la couche de sortie, Φ est une fonction d'activation, H, O et R sont respectivement les poids à l'entrée de la couche cachée, de sortie et récurrente (pour simplifier la lecture des formules, nous avons omis les biais). La sortie $\mathbf{y_t}$ est un vecteur qui représente une distribution de probabilités sur l'ensemble des étiquettes. À partir de celle-ci nous pouvons calculer la prédiction du modèle en sélectionnant l'étiquette correspondant à la position avec la probabilité maximale dans $\mathbf{y_t}$. $\mathbf{h_{t-1}}$ est la couche cachée calculée à l'étape de traitement précédente prise en compte comme contexte par le réseau d'Elman, alors que $\mathbf{y_{t-1}}$ est la couche de sortie de l'étape précédente et est utilisé comme contexte à l'étape actuelle par le réseau de Jordan. $\mathbf{I_t}$ est l'entrée du réseau, constituée généralement de la concaténation des plongements des mots dans une fenêtre de taille d_w (de ***win**dow **w**ord*) fixée autour du mot courant w_t. Nous indiquons avec $E_w(w_i)$ le plongement d'un mot quelconque w_i, sélectionné dans la matrice E_w. $\mathbf{I_t}$ est alors défini comme suit :

$$\mathbf{I_t} = [E_w(w_{t-d_w})...E_w(w_t)...E_w(w_{t+d_w})] \quad [4]$$

Les crochets [ ] indiquent la concaténation de vecteurs (ou matrices dans la suite de l'article). La fonction $softmax$, calculée sur un ensemble S, de taille m, de valeurs numériques v_i, associées à des éléments discrets $i \in [1, m]$ (dans notre cas, les index des étiquettes), produit la probabilité associée à chaque élément de la façon suivante :

$$\forall i \in [1, m]\ p(i) = \frac{e^{v_i}}{\sum_{j=1}^{m} e^{v_j}}$$

2.2. *Réseaux* Long Short-Term Memory *(LSTM)*

Bien que le nom LSTM soit utilisé pour indiquer un réseau neuronal à part entière, il définit seulement un type de couche cachée. Dans LSTM, la sortie de la couche cachée est contrôlée par des unités dites *gates* et par une unité de mémoire dite *cell state*. Ces unités déterminent la façon dont l'information passée est prise en compte pour calculer la sortie de la couche cachée à l'étape présente. En particulier, les unités *forget*, *input* et *cell state* sont calculées comme suit :

$$\mathbf{f_t} = \Phi(W_f \mathbf{h_{t-1}} + U_f \mathbf{I_t}) \quad [5]$$

$$\mathbf{i_t} = \Phi(W_i \mathbf{h_{t-1}} + U_i \mathbf{I_t}) \quad [6]$$

$$\hat{\mathbf{c}}_\mathbf{t} = \Gamma(W_c \mathbf{h_{t-1}} + U_c \mathbf{I_t}) \quad [7]$$

Γ est une autre fonction d'activation[1], $\hat{\mathbf{c}}_\mathbf{t}$ est une valeur intermédiaire utilisée pour mettre à jour la valeur de l'unité *cell state* de la façon suivante :

$$\mathbf{c_t} = \mathbf{f_t} \odot \mathbf{c_{t-1}} + \mathbf{i_t} \odot \hat{\mathbf{c}}_\mathbf{t} \quad [8]$$

$\odot$ est la multiplication élément par élément. Une fois que toutes ces quantités ont été calculées, la valeur de l'unité *output* est calculée et utilisée pour produire la sortie de la couche cachée à l'étape présente t :

$$\mathbf{o_t} = \Phi(W_o \mathbf{h_{t-1}} + U_o \mathbf{I_t}) \quad [9]$$

$$\mathbf{h_t}^{\text{LSTM}} = \mathbf{o_t} \odot \Phi(\mathbf{c_t}) \quad [10]$$

Comme nous pouvons le voir, chaque *gate* et la *cell state* ont leurs matrices de paramètres W et U. L'évolution de la couche LSTM nommée GRU (de *Gated Recurrent Units*) (Cho *et al.*, 2014), combine les unités *forget* et *input* en une seule unité, et combine également la couche cachée précédente avec l'unité *cell state* :

$$\mathbf{z_t} = \Phi(W_z \mathbf{h_{t-1}} + U_z \mathbf{I_t}) \quad [11]$$

$$\mathbf{r_t} = \Phi(W_r \mathbf{h_{t-1}} + U_r \mathbf{I_t}) \quad [12]$$

$$\hat{\mathbf{h_t}} = \Gamma(W(\mathbf{r_t} \odot \mathbf{h_{t-1}}) + U\mathbf{I_t}) \quad [13]$$

$$\mathbf{h_t}^{\text{GRU}} = (1 - \mathbf{z_t}) \odot \mathbf{h_{t-1}} + \mathbf{z_t} \odot \hat{\mathbf{h_t}} \quad [14]$$

Les schémas des couches cachées LSTM et GRU sont montrés en figure 2.

1. Dans la littérature, dans les réseaux LSTM et GRU Φ et Γ sont respectivement la sigmoïde et la tangente hyperbolique.

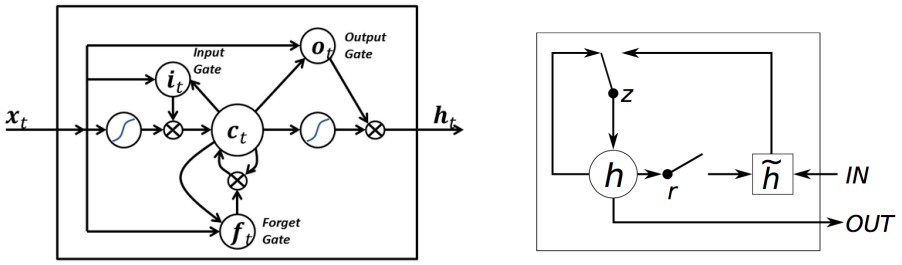


Figure 2. *Schémas des couches cachées LSTM et GRU*

2.3. *LD-RNN : RNN modélisant les dépendances entre étiquettes*

La variante de RNN que nous proposons peut être vue comme un RNN ayant une connexion récurrente entre la couche de sortie et la couche d'entrée. Cette modification permet 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 correspondance qui sert à transformer les représentations creuses *one-hot*[2] en représentations distributionnelles. Ces dernières peuvent encoder 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 des 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 ainsi exploiter de la même façon des arbres syntaxiques ou des entités nommées étendues. L'idée d'utiliser des plongements pour les étiquettes n'est pas nouvelle en soi : Chen et Manning (2014) l'ont introduite dans le contexte de l'analyse syntaxique en dépendances, ce qui a donné un analyseur très performant. Dans ce travail nous allons au-delà de cette approche. En plus de pré-apprendre les plongements des étiquettes, nous utilisons plusieurs étiquettes comme contexte. Cela permet d'apprendre d'une façon très efficace les dépendances entre étiquettes. Pour cette raison, nous appelons cette variante de RNN *LD-RNN*, pour *Label Dependency aware RNN*.

Avec le même formalisme que précédemment, nous notons E_w la matrice des plongements des mots, et E_l celle des plongements des étiquettes. $\mathbf{I_t}$ est la même entrée introduite plus haut, alors que :

$$\mathbf{L_t} = [E_l(y_{t-d_l+1})...E_l(y_{t-d_l+2})...E_l(y_{t-1})] \quad [15]$$

est la concaténation des vecteurs qui représentent les étiquettes prédites aux d_l positions précédentes. Nous notons que y_t indique ici l'étiquette prédite par le modèle,

2. La représentation *one-hot* d'un *token* représenté par son index i dans un dictionnaire est un vecteur v de la même taille que le dictionnaire, et qui vaut zéro partout sauf à la position i où il vaut 1.

et non pas la distribution de probabilités $\mathbf{y_t}$ calculée par le *softmax*. La sortie de la couche cachée est alors calculée comme suit :

$$\mathbf{h_t}^{\text{LD-RNN}} = \Phi(H\ [\mathbf{I_t}\mathbf{L_t}]) \quad [16]$$

Les autres couches sont calculées comme précédemment.

Contrairement à ce qu'indiquaient Dinarelli et Tellier (2016a), la concaténation des deux entrées, mots et étiquettes, à la place de leur somme, ne change pas la capacité du modèle, $H[\mathbf{I_t}\mathbf{L_t}]$ étant équivalent à $H_w\mathbf{I_t} + H_l\mathbf{L_t}$ si $H = [H_w H_l]$.

Les plongements d'étiquettes encodent les caractéristiques du contexte, c'est-à-dire les étiquettes cooccurrentes avec une étiquette donnée. De ce fait, ces plongements encodent l'information permettant de prédire des successions d'étiquettes cohérentes. Les autres RNN n'utilisent pas les étiquettes précédentes dans leur fonction de décision, ou ils utilisent une représentation peu efficace des celles-ci. Ces RNN peuvent donc produire des séquences d'étiquettes incohérentes. Cette limitation a conduit aux modèles hybrides *RNN+CRF* (Huang *et al.*, 2015 ; Lample *et al.*, 2016 ; Ma et Hovy, 2016).

Une conséquence importante de l'utilisation des plongements d'étiquettes est une robustesse accrue du modèle par rapport aux erreurs de prédiction. Puisque nous utilisons plusieurs étiquettes prédites comme contexte (voir $\mathbf{L_t}$ plus haut), en phase de test il est assez improbable que plusieurs erreurs de prédiction subsistent dans le même contexte. Même dans ce cas, grâce aux propriétés des plongements, les étiquettes erronées ont des représentations assez proches de celles des étiquettes attendues. Reprenons un exemple cité dans Mikolov *et al.* (2013b) : si l'on remplace *Paris* par *Rome* dans un texte, cela n'a aucun effet dans plusieurs tâches : ce sont en effet tous les deux des noms propres pour un étiquetage POS, des lieux pour une extraction d'entités nommées, etc. L'utilisation de plongements d'étiquettes permet d'avoir la même robustesse sur les étiquettes.

Il ne se passe en général pas la même chose avec un modèle de Jordan classique. Dans ce modèle, les étiquettes sont représentées soit par la distribution de probabilités sur les étiquettes calculée avec le $softmax$, soit par la représentation *one-hot* calculée à partir de celle-ci. Ces deux représentations sont très creuses[3], elles donnent donc au modèle bien moins de souplesse par rapport à ce que l'on peut obtenir avec des plongements d'étiquettes.

D'un autre point de vue, on peut interpréter le calcul de la couche cachée dans un réseau de Jordan comme sélectionnant des plongements de la matrice R (voir formule 2). Puisque $\mathbf{y_{t-1}}$ est un vecteur creux, nous pouvons interpréter la multiplication de $\mathbf{y_{t-1}}$ par la matrice R comme la sélection d'un plongement pour l'étiquette correspondante. Même avec cette interprétation, il y a une différence importante entre le réseau de Jordan et notre variante. En effet, une fois que la sélection du plongement

3. Nous avons constaté expérimentalement que dans la sortie du *softmax* la masse des probabilités est concentrée dans une ou très peu d'étiquettes seulement (2 ou 3 au maximum).

pour l'étiquette a été faite avec $R\mathbf{y_{t-1}}$, le résultat n'est pas directement utilisé dans la transformation linéaire opérée par la matrice H. Seul le contexte au niveau des mots est multiplié par H ($H\mathbf{I_t}$). Le résultat de cette multiplication est additionné à $R\mathbf{y_{t-1}}$ pour produire l'entrée finale de la couche cachée.

Dans notre variante en revanche, nous sélectionnons d'abord le plongement de chaque étiquette avec $E[y_i]$[4]. Ensuite nous appliquons la transformation linéaire de la couche cachée sur les mots et sur les étiquettes ($H[\mathbf{I_t}\mathbf{L_t}]$). Dans notre variante donc les étiquettes subissent toujours deux transformations : i) la conversion de *one-hot* en plongements, ii) la transformation linéaire de la couche cachée.

2.4. *Apprentissage et inférence*

L'apprentissage de notre variante est réalisée en minimisant l'entropie croisée entre l'étiquette attendue e_t et la sortie du réseau $\mathbf{y_t}$ à la position t dans une séquence, plus un terme de régularisation $L2$:

$$C = -e_t \odot log(\mathbf{y_t}) + \frac{\lambda}{2} |\Theta|^2 \quad [17]$$

où λ est un hyperparamètre du modèle, Θ est un raccourci pour les paramètres d'un modèle. Nous désignons par e_t la représentation creuse *one-hot* de l'étiquette attendue. Puisque $\mathbf{y_t}$ est une distribution de probabilités sur les étiquettes, nous interprétons la sortie du réseau comme la probabilité $P(i|\mathbf{I_t}, \mathbf{L_t})\ \forall i \in [1, m]$, $\mathbf{I_t}$ et $\mathbf{L_t}$ étant les entrées du réseau (mots et étiquettes), i étant l'index d'une des m étiquettes définies dans la tâche. Nous associons au modèle *LD-RNN* la fonction de décision suivante :

$$argmax_{i\in[1,m]} P(i|\mathbf{I_t}, \mathbf{L_t}) \quad [18]$$

Nous pouvons voir que cette fonction de décision est locale, puisque la probabilité des étiquettes est normalisée à chaque position de la séquence traitée. Malgré tout, grâce à l'utilisation des plongements d'étiquettes $\mathbf{L_t}$, la variante *LD-RNN* arrive à modéliser et prendre en compte les dépendances entre étiquettes. Puisque les autres RNN tels que le modèle d'Elman et les LSTM n'utilisent pas les étiquettes comme information contextuelle, leur fonction de décision peut être définie comme :

$$argmax_{i\in[1,m]} P(i|\mathbf{I_t}) \quad [19]$$

Cette fonction de décision peut conduire à des séquences de prédictions incohérentes.

Nous utilisons l'algorithme de rétropropagation classique avec vitesse (*momentum*) et la descente de gradient pour apprendre les poids du modèle (Bengio, 2012), le préférant à l'algorithme de rétropropagation à travers le temps (*Back-propagation through time BPTT*) (Werbos, 1990). En effet, étant donné leur architecture récurrente, pour apprendre proprement un RNN, il faudrait utiliser l'algorithme BPTT. Cet algorithme consiste à *dérouler* (*unfold*) un RNN sur N étapes passées, N étant un paramètre, et utilisant donc les N entrées et contextes précédents pour mettre à jour

4. Dans ce cas, le vecteur $\mathbf{y_i}$ est explicitement converti en une représentation *one-hot*, qui est équivalente à un index scalaire y_i.

tous les paramètres du modèle. L'algorithme de rétropropagation classique est alors appliqué. Cela équivaut donc à apprendre un réseau de profondeur N.

L'algorithme BPTT est censé permettre d'apprendre des contextes arbitrairement longs. Cependant, Mikolov *et al.* (2011) ont montré que les RNN pour les modèles de langues apprennent mieux avec seulement $N = 5$ 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 la plupart des tâches de TAL, garder en mémoire un contexte 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 rétropropagation à travers le temps est plus coûteux que l'algorithme classique, Mesnil *et al.* (2013) ont préféré utiliser explicitement l'information contextuelle donnée à la connexion récurrente dans les RNN, et utiliser l'algorithme de rétropropagation classique, apparemment sans perte de performances. Dans cet article, nous adoptons la même stratégie d'apprentissage.

Lorsque l'on utilise explicitement les étiquettes prédites aux étapes précédentes dans un réseau de Jordan, la sortie de la couche cachée est calculée par :

$$\mathbf{h_t} = \Phi(R[\mathbf{y_{t-d_l+1}}\ \mathbf{y_{t-d_l+2}} \cdots \mathbf{y_{t-1}}] + H\ \mathbf{I_t}) \quad [20]$$

où d_l est la taille du contexte côté sortie utilisée explicitement dans le réseau. Une modification équivalente permet de prendre en compte les couches cachées précédentes dans un RNN d'Elman, LSTM et GRU.

2.5. *Vers des RNN plus sophistiqués*

2.5.1. *Convolution sur les caractères*

Même si les plongements fournissent une représentation très riche des mots, plusieurs travaux sur les RNN, par exemple (Huang *et al.*, 2015 ; Chiu et Nichols, 2015 ; Lample *et al.*, 2016 ; Ma et Hovy, 2016), ont montré que des modèles plus efficaces peuvent être obtenus en utilisant une couche de convolution sur les caractères de chaque mot donné en entrée au réseau. Cette information est en effet très importante pour permettre au modèle de généraliser les mots dont certaines formes fléchies sont rares ou même hors vocabulaire en phase de test, les plongements sur les mots étant bien moins efficaces dans ces cas. La convolution sur les caractères offre en plus l'avantage d'être assez générique : elle peut être appliquée exactement de la même façon à des langues différentes, permettant la réutilisation d'un même système sur des tâches différentes.

Dans cet article nous utilisons une couche de convolution sur les caractères comme celle utilisée dans (Collobert *et al.*, 2011) pour les mots. Formellement, pour un mot w de longueur $|w|$, en indiquant par $E_{ch}(w, i)$ le plongement du caractère à la position i du mot w, une convolution de taille $2d_c + 1$ est appliquée comme suit :

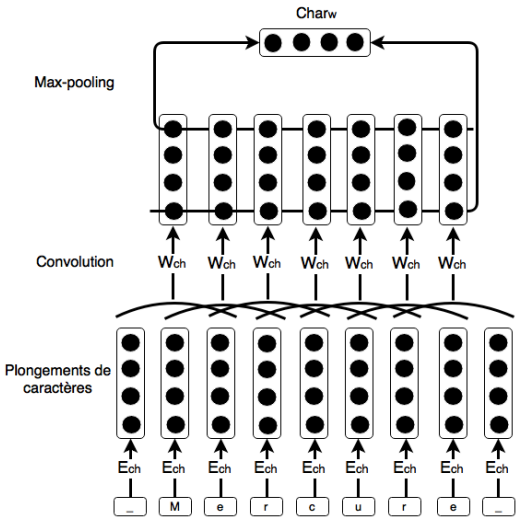


Figure 3. *Schéma de la couche de convolution appliquée sur les caractères du mot* Mercure

$$\forall i \in [1, |w|]\ Conv_i = W_{ch}[E_{ch}(w, i - d_c)...E_{ch}(w, i)...E_{ch}(w, i + d_c)] \quad [21]$$

$$Char_w = Max([Conv_1 \dots Conv_{|w|}]) \quad [22]$$

La sortie de la convolution $[Conv_1 \dots Conv_{|w|}]$ est la concaténation du résultat de l'application du modèle linéaire W_{ch} à chaque fenêtre de taille $2d_c+1$ de plongements de caractères. Pour que la convolution soit bien définie aussi pour les premiers et derniers caractères de chaque mot, celui-ci est augmenté par un préfixe et un suffixe de taille d_c, obtenus par la concaténation d'un caractère conventionnel (nous avons choisi le caractère "_"). Un schéma de la convolution sur les caractères est montré en figure 3.

La taille de la matrice $[Conv_1 \dots Conv_{|w|}]$ est $|C| \times |w|$, $|C|$ étant la taille de la couche de convolution. La fonction Max, utilisée pour calculer le *Max-pooling* (Collobert *et al.*, 2011), calcule les maximums dans la direction de la longueur du mot, produisant une sortie de la taille $|C|$, indépendante donc de la longueur du mot. La sortie finale du *Max-pooling* $Char_w$ peut être interprétée comme une représentation distributionnelle du mot w encodant l'information au niveau des ses caractères. Celle-ci est une information complémentaire par rapport aux plongements des mots, qui encodent des informations intermots. En effet, elle fournit au modèle des informations similaires à celles fournies par des traits discrets tels que les préfixes, les suffixes, l'information de capitalisation, etc.

2.5.2. *Couches cachées* ReLU

Depuis quelques années, les couches cachées des réseaux neuronaux utilisent les unités *ReLU* (Bengio, 2012). Celles-ci ont remplacé les unités utilisant des fonctions d'activation telles que la sigmoïde. Ce type de fonction, appelée *squashing function*[5], a tendance à saturer après un certain nombre d'itérations d'apprentissage, ne permettant pas l'apprentissage de réseaux arbitrairement grands et/ou profonds.

Les unités *ReLU* utilisent une fonction d'activation linéaire rectifiée, définie comme $f(x) = max(0, x)$. Malgré leur efficacité reconnue, les unités *ReLU* ne transmettent pas de signal lorsque l'entrée est négative. Puisque les paramètres des différentes couches des réseaux neuronaux sont repartis autour de zéro, cela implique que les unités *ReLU* n'utilisent pas l'information d'environ la moitié de leur entrée.

Pour pallier cette limitation, des variantes des unités *ReLU* ont été proposées (He *et al.*, 2015). Une première, appelée *Leaky ReLU* (LReLU), est définie comme :

$$LReLU(x) = max(0, x) + \epsilon\, min(0, x)$$

Cette version garde la partie négative de l'entrée, mais celle-ci est multipliée par une valeur négligeable ϵ fixée (He *et al.* (2015) proposent $0,01$ ou $0,001$ pour ϵ).

Une autre variante proposée par He *et al.* (2015) est une généralisation de *Leaky ReLU* dans laquelle ϵ est remplacé par une matrice de paramètres A. Cette version est appelée *ReLU* paramétré (PReLU) :

$$PReLU(x) = max(0, x) + A \odot min(0, x)$$

Elle a l'avantage d'utiliser un facteur multiplicatif paramétrique pour chaque valeur de l'entrée x, mais ajoute au modèle les paramètres A qui sont appris en même temps que les autres. Grâce à cette version, He *et al.* (2015) obtiennent des résultats meilleurs que l'évaluation humaine sur une tâche de classification d'images.

Nous avons testé toutes ces variantes d'unités *ReLU* dans notre variante *LD-RNN*.

2.6. *Complexité des RNN*

La modélisation des dépendances entre étiquettes, plus efficace dans notre variante *LD-RNN*, est obtenue au prix de l'utilisation de plus de paramètres par rapport aux RNN simples. Cependant le nombre de paramètres est bien moins important que dans les réseaux LSTM et GRU. Dans cette section, nous faisons une analyse de la complexité de chaque type de RNN en termes du nombre de paramètres.

Nous introduisons les symboles suivants : $|H|$ et $|O|$ sont respectivement les tailles des couches cachée et de sortie. La taille de la couche de sortie correspond au nombre d'étiquettes. N est la taille des plongements, la même pour les mots et les étiquettes.

5. En effet cette fonction met en correspondance l'ensemble $\mathbb{R}$ avec l'intervalle réel $[0, 1]$. En théorie cela ne pose aucun problème puisque entre 0 et 1 il y a un nombre infini de réels, mais sur un ordinateur ce n'est pas le cas.

d_w et d_l sont les tailles des contextes déjà introduites précédemment. Nous analysons la couche des plongements et la couche cachée, les autres couches ont le même nombre de paramètres dans tous les réseaux.

Dans la couche cachée des RNN d'Elman et de Jordan nous avons les paramètres suivants :

$$\{|H| * |H|\}_R + \{|H| * (2d_w + 1)N\}_{H^{\text{Elman}}}$$

$$\{|O| * |H|\}_R + \{|H| * (2d_w + 1)N\}_{H^{\text{Jordan}}}$$

Les indices indiquent de quelle matrice proviennent les paramètres. Le facteur $(2d_w + 1)N$ vient de l'utilisation des plongements de $(2d_w + 1)$ mots en entrée, alors que le facteur $|O| * |H|$ dans le réseau de Jordan est dû au fait que R relie la couche de sortie à la couche cachée.

Dans le réseau *LD-RNN* nous avons :

$$\{\{((2d_w + 1)N + d_l N) * |H| + |O| * N\}_{E_l}\}_{H^{\text{LD-RNN}}}$$

Le facteur $|O| * N$ vient de la matrice E_l contenant $|O|$ plongements d'étiquettes de taille N. Dans les tâches utilisées dans cet article pour l'évaluation, nous avons choisi $|H| > |O|$ et $N = |H|$. Par conséquent, l'utilisation des plongements d'étiquettes dans *LD-RNN* n'augmente pas en soi le nombre de paramètres.

En revanche, la matrice H dans *LD-RNN* est dimensionnée pour connecter tous les plongements de mots et d'étiquettes à chaque neurone de la couche cachée. $H^{\text{LD-RNN}}$ contient donc $d_l N$ paramètres de plus que H^{Elman} et H^{Jordan}.

Dans les réseaux LSTM et GRU nous avons 2 matrices additionnelles W et U, pour chaque *gate* et pour la *cell state*, utilisées pour connecter la couche cachée précédente et l'entrée courante. Ces 2 matrices contiennent respectivement $|H| * |H|$ et $(2d_w + 1)N * |H|$ paramètres.

Utilisant la même notation que précédemment, pour LSTM et GRU nous avons :

$$\{4(|H| * |H| + |U| * (2d_w + 1)N)\}_{H^{\text{LSTM}}}$$

$$\{3(|H| * |H| + |U| * (2d_w + 1)N)\}_{H^{\text{GRU}}}$$

Le 3 pour le RNN GRU est dû au fait que dans ce réseau nous avons 2 unités *gate*, plus l'unité *cell state*.

En conclusion, nous pouvons constater que la variante *LD-RNN* utilise plus de paramètres que les RNN simples, mais bien moins que les réseaux plus sophistiqués LSTM et GRU.

2.7. *Réseaux* forward, backward *et bidirectionnels*

Nous avons étudié le comportement de notre réseau *LD-RNN* dans trois versions : *forward*, *backward* et bidirectionnel (Schuster et 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 de la fin vers le début.

Les RNN bidirectionnels utilisent à la fois l'information contextuelle passée et future pour prédire la prochaine étiquette. Les informations futures ne sont pas disponibles directement, c'est pourquoi un modèle bidirectionnel se sert d'un modèle *backward* comme modèle auxiliaire. Ce modèle est utilisé dans une première passe en arrière (*backward*) pour prédire les informations futures. Ensuite le modèle bidirectionnel fait une passe en avant, utilisant l'information future prédite par le modèle *backward* comme contexte.

Comme cela est décrit en détail par Schuster et Paliwal (1997), il existe deux versions de modèle bidirectionnel. La première utilise un modèle *forward* et un modèle *backward* séparés. La seconde est constituée d'un modèle unique, dans lequel on distingue des couches *forward* et des couches *backward*.

Dans les premières phases de test de nos réseaux, nous avons observé que les deux versions sont équivalentes sur les tâches présentées dans cet article. Nous avons donc choisi d'utiliser la première version. Celle-ci présente l'avantage de pouvoir être initialisée totalement avec les poids des modèles *forward* et *backward* déjà calculés. Ceci a pour conséquence que le modèle bidirectionnel est très proche de l'optimum dès la première itération, et très peu d'itérations suffisent donc pour obtenir un modèle très performant.

Dans cette version de modèle bidirectionnel la sortie est calculée comme la moyenne géométrique des sorties des modèles *forward* et *backward* :

$$\mathbf{y_t} = \sqrt{\mathbf{y_t^f} \odot \mathbf{y_t^b}} \qquad [23]$$

$\mathbf{y_t^f}$ et $\mathbf{y_t^b}$ étant respectivement les sorties du modèle *forward* et *backward*.

3. Évaluation

3.1. *Corpus pour la compréhension automatique de la parole*

Nous avons évalué nos modèles sur deux tâches assez populaires de compréhension automatique de la parole.

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 fondée 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.

ATIS est une tâche assez simple qui date de 1993. Les données d'apprentissage sont constituées de 4 978 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 des corpus `ATIS-3 NOV93` et `DEC94`. Il n'y a pas de données de développement, 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 1 117 et 85 unités, respectivement. Nous utilisons la version du corpus publiée dans Raymond et Riccardi (2007), dans laquelle sont aussi disponibles des classes de mots, comme par exemple les noms des villes, des aéroports, expressions de temps etc., permettant aux modèles de généraliser certains mots rares ou absents en phase de test. Nous renvoyons à Dahl *et al.* (1994) pour plus de détails. Dans cet article nous nous comparons aux travaux qui ont utilisé la version officielle d'ATIS (Raymond et Riccardi, 2007).

Un exemple de phrase de ce corpus est *« I want all the flights from Boston to Philadelphia today »*[6]. 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*, ils sont donc annotés avec l'étiquette "vide" *O*. Ces caractéristiques de l'annotation montrent la simplicité de cette tâche : l'annotation est assez creuse, seulement 3 mots de la phrase sont annotés avec des concepts non vides. Chaque étiquette correspond à un seul mot, il n'y a aucune segmentation des étiquettes sur plusieurs mots (comme c'est le cas avec un codage *BIO*). À cause de ces caractéristiques, la tâche ATIS est similaire à une tâche d'étiquetage en partie du discours (*POS tagging*), où il n'y a pas non plus de segmentation des étiquettes sur les mots. Cette tâche peut aussi être comparée à de la reconnaissance d'entités nommées linéaires, où l'annotation est également creuse.

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 1 250 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 riche. Des composants sémantiques peuvent être combinés pour former des étiquettes sémantiques complexes.[7] Outre la richesse de l'annotation utilisée, une autre source de difficultés pour l'étiquetage provient des phénomènes de coréférence. Certains mots ne peuvent pas être annotés correctement sans connaître un contexte relativement long, comprenant souvent le tour de parole précédent. 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 2.

La tâche MEDIA peut être modélisée comme un étiquetage de séquences en utilisant la convention classique *BIO* (Ramshaw et Marcus, 1995).

Grâce à ces différentes caractéristiques, combinées avec une taille relativement réduite qui permet l'apprentissage des modèles en un temps raisonnable, ces deux tâches constituent un terrain idéal pour l'évaluation de modèles pour l'étiquetage de séquences. Un exemple comparatif de ces deux tâches est montré dans le tableau 1.

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

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

MEDIA			ATIS		
Mots	**Classes**	**Étiquettes**	**Mots**	**Classes**	**Étiquettes**
Oui	-	Answer-B	i'd	-	O
l'	-	BDObject-B	like	-	O
hotel	-	BDObject-I	to	-	O
le	-	Object-B	fly	-	O
prix	-	Object-I	Delta	airline	airline-name
à	-	Comp.-payment-B	between	-	O
moins	relative	Comp.-payment-I	Boston	city	fromloc.city-name
cinquante	tens	Paym.-amount-B	and	-	O
cinq	units	Paym.-amount-I	Chicago	city	toloc.city-name
euros	currency	Paym.-currency-B			

Tableau 1. *Un exemple de phrase annotée pris des deux corpus MEDIA et ATIS*

	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

Tableau 2. *Statistiques sur le corpus MEDIA*

3.2. *Choix architecturaux et réglages*

Notre variante de RNN *LD-RNN* a été implémentée par nous-mêmes en *Octave*[8] avec le support de la bibliothèque *OpenBLAS*.[9]

Les modèles *LD-RNN* sont entraînés suivant la procédure suivante :

– deux modèles de langage neuronaux comme ceux de Bengio *et al.* (2003) sont entraînés pour générer les plongements des mots et des étiquettes ;

– les modèles *forward* et *backward* sont appris en utilisant les plongements entraînés à l'étape précédente ;

– le modèle bidirectionnel est appris en utilisant comme poids initiaux les modèles *forward* et *backward* entraînés à l'étape précédente.

Les plongements des mots et des étiquettes sont appris sur les données d'apprentissage de la tâche concernée. Nos systèmes sont donc entièrement endogènes. Nous avons également fait des expériences avec des plongements appris avec *word2vec* (Mikolov *et al.*, 2013a). Les résultats obtenus ne sont pas significativement différents des autres, ils ne seront donc pas discutés dans la suite de l'article.

8. https ://www.gnu.org/software/octave/ ; Notre logiciel est décrit à la page http ://www.marcodinarelli.it/software.php et il est disponible sous requête.
9. http ://www.openblas.net ; Cette bibliothèque permet une accélération d'un facteur d'environ 330 par rapport à une bibliothèque BLAS basique, en utilisant 16 cœurs.

Nous avons grossièrement optimisé le nombre d'époques d'entraînement pour chaque modèle sur les données de développement : 30 époques pour entraîner les plongements des mots, 20 pour les plongements d'étiquettes ; 30 époques aussi pour les modèles *forward* et *backward*, 8 pour le modèle bidirectionnel (l'optimum de ce dernier est souvent atteint à la première époque sur ATIS, entre la 3^e et la 5^e sur MEDIA). À la fin de l'entraînement, nous gardons le modèle qui donne la meilleure précision sur les données de développement, nous arrêtons l'apprentissage si elle n'est pas améliorée pendant 5 époques consécutives (*Early stopping* (Bengio, 2012)).

Nous initialisons les poids des modèles avec la procédure dite *Xavier initialization* (Bengio, 2012). Celle-ci est motivée théoriquement dans (He *et al.*, 2015) comme permettant de préserver pendant l'apprentissage la variance de la distribution dans laquelle les valeurs initiales des poids sont prises.

Nous avons optimisé certains hyperparamètres de nos modèles sur les données de développement : sur la base de cette optimisation, le taux d'apprentissage est initialisé à $0,5$ et diminué d'une valeur fixe à chaque époque (*Learing Rate decay*). Cette valeur est le rapport entre le taux d'apprentissage initial et le nombre d'itérations. Pour la régularisation, nous combinons le *dropout* et la régularisation L_2 (Bengio, 2012), la meilleure probabilité de *dropout* sur la couche cachée est de $0,5$, alors qu'elle est de $0,2$ sur les plongements pour ATIS, $0,15$ pour MEDIA. Le meilleur coefficient pour la régularisation L_2 est de $0,01$ dans les modèles *forward* et *backward* et il est de $3e^{-4}$ pour le modèle bidirectionnel.

Nous avons aussi mené des expériences pour choisir une taille optimale pour les couches des plongements, les couches cachées et la couche de convolution. Pour rendre la phase d'optimisation la plus rapide possible, nous avons optimisé la taille de ces couches sur les données de développement des deux corpus, avec le seul modèle *forward*, intégrant seulement les mots et les étiquettes comme informations d'entrée (donc sans classes ni convolution sur les caractères). La meilleure taille trouvée pour les plongements et la couche cachée a été de 200 pour les deux tâches. La meilleure taille pour la couche de convolution est de 50 pour la tâche ATIS, et de 80 pour MEDIA. Dans les deux cas, la meilleure taille pour la fenêtre de convolution est de 1. Une fenêtre de taille 3 (un caractère à gauche et un à droite, plus le caractère central) donne à peu près les mêmes résultats, nous préférons donc le modèle le plus simple. Avec une fenêtre de taille 5 les résultats commencent à se dégrader. Nous avons également optimisé la taille du contexte de mots et d'étiquettes dans notre variante de RNN *LD-RNN*. Sur ATIS, les meilleures tailles pour ces contextes sont de 11 pour les mots (5 à gauche, 5 à droite et le mot courant) et de 5 pour les étiquettes. Sur MEDIA elles sont de 7 et 5 respectivement. Dans le cas des étiquettes, le contexte est pris d'un seul côté par rapport au mot courant. Seul le modèle bidirectionnel utilise un contexte de deux côtés. Par conséquent notre modèle bidirectionnel utilise un contexte de 10 étiquettes : 5 avant et 5 après le mot courant.

Tous les paramètres fixés dans cette phase d'optimisation ont été utilisés pour obtenir des modèles *baseline*. Le but est de comprendre le comportement des différents réseaux en ajoutant les autres informations d'entrée : les classes disponibles dans les

Modèle	Mesure F1		
	Forward	*Backward*	Bidirectionnel
LD-RNN Mots	94,23	94,30	94,45
LD-RNN Mots + CC	94,56	94,69	94,79
LD-RNN Mots + Classes	95,31	95,42	95,53
LD-RNN Mots + Classes + CC	**95,55**	**95,45**	**95,65**
$LD\text{-}RNN_{LReLU}$ Mots + Classes + CC	95,29	**95,45**	95,52
$LD\text{-}RNN_{PReLU}$ Mots + Classes + CC	94,67	95,08	95,13

Tableau 3. *Résultats (F1) sur le corpus ATIS selon différents niveaux d'information*

deux tâches et la convolution sur les caractères. Comme cela sera discuté plus loin, certains paramètres ont été ensuite raffinés.

En ce qui concerne le temps d'apprentissage de nos modèles, le temps total pour apprendre et tester les trois modèles *forward*, *backward* et bidirectionnel, utilisant seulement les mots et les étiquettes comme entrées, est d'environ 1 h 10 pour la tâche MEDIA, 40 minutes pour ATIS. Ce temps s'élève à 2 h 00 pour MEDIA, à 2 h 10 pour ATIS, en utilisant aussi les classes et la convolution sur les caractères. Tous ces temps sont mesurés sur un processeur *Intel Xeon E5-2620* à 2.1 *GHz* en utilisant 16 cœurs.

3.3. *Résultats*

Tous les résultats discutés dans cette section sont obtenus en moyennant 6 expériences différentes, dans lesquelles les plongements sont entraînés une fois pour toutes, alors que les poids des autres couches sont réinitialisés à chaque fois avec une graine différente.

3.3.1. *Résultats principaux avec différents niveaux d'information*

Nous commentons ici les résultats obtenus en ajoutant progressivement les différentes informations disponibles pour l'entrée de nos modèles : les mots seuls, les mots plus la convolution sur les caractères, les mots plus les classes, les mots plus les classes et la convolution. Ces différents niveaux d'information sont indiqués par *Mots*, *Classes* et *CC* (pour convolution de caractères). Nos modèles utilisent tous un contexte d'étiquettes de taille 5, la taille optimale déjà commentée dans la section 3.2.

Les résultats obtenus pour la tâche ATIS sont donnés dans le tableau 3, et ceux sur MEDIA dans le tableau 4. Nous y montrons également les résultats obtenus avec les différentes couches cachées *ReLU* décrites dans la section 2.5.2. Par défaut nous utilisons la couche *ReLU* classique, l'utilisation des couches *Leaky ReLU* et *ReLU* paramétré est indiquée dans le nom du modèle avec *LReLU* et *PReLU*, respectivement.

Comme nous pouvons le voir dans les tableaux, le comportement des modèles sur les deux tâches est assez similaire. Sur la tâche ATIS notamment, en ajoutant les différents niveaux d'information les résultats s'améliorent progressivement. Les meilleurs résultats sont obtenus avec tous les niveaux d'information (mots, classes et convo-

Modèle	Mesure F1		
	Forward	*Backward*	Bidirectionnel
LD-RNN Mots	85,39	86,54	87,05
LD-RNN Mots + CC	85,41	86,48	86,98
LD-RNN Mots + Classes	**85,46**	86,59	87,16
LD-RNN Mots + Classes + CC	85,38	**86,79**	**87,22**
$LD\text{-}RNN_{LReLU}$ Mots + Classes + CC	**85,48**	86,72	87,18
$LD\text{-}RNN_{PReLU}$ Mots + Classes + CC	84,91	86,36	86,79

Tableau 4. *Résultats (F1) sur le corpus MEDIA selon différents niveaux d'information*

lution des caractères) et ce, malgré que certains gains ne semblent pas significatifs compte tenu de la taille assez réduite de cette tâche.

Cela est plus évident sur la tâche MEDIA, où l'ajout de la convolution des caractères aux mots provoque des légères pertes de performances. Pour les deux tâches, nous avons analysé le comportement du modèle lors de l'apprentissage et nous avons constaté que le modèle était saturé. C'est-à-dire que la taille de la couche cachée, optimisée avec le modèle *forward* et en n'utilisant que les mots, semblait ne pouvoir plus modéliser de façon efficace l'information supplémentaire donnée en entrée au modèle. Nous avons alors lancé d'autres expériences avec une couche cachée de taille 256, qui ont donné les résultats montrés dans les tableaux avec le modèle *LD-RNN Mots + Classes + CC*. Par manque de temps nous n'avons pas optimisé davantage les paramètres de ces réseaux.

Nous notons que l'utilisation des couches cachées *LReLU* et *PReLU* n'améliore pas les résultats obtenus avec la couche *ReLU* classique, voire les dégrade dans le cas de la couche *PReLU*. Ces résultats relativement décevants ont une explication : en analysant la phase d'apprentissage nous avons remarqué que les valeurs de la fonction de coût (l'entropie croisée) sur les données d'apprentissage sont meilleures qu'avec la couche *ReLU* classique. Il s'agit donc de surapprentissage. Avec la couche *PReLU*, notamment, en utilisant les mêmes paramètres, le réseau est instable (le fameux problème du *exploding gradient* (Hochreiter *et al.*, 2001)). Nous avons résolu le problème en baissant le taux d'apprentissage de $0,5$ à $0,1$.

Malgré un taux d'apprentissage bien moins important, ces couches cachées conduisent à des valeurs de la fonction de coût sur les données d'apprentissage bien meilleures qu'avec une couche cachée *ReLU* classique. Cela montre un grand potentiel pour apprendre les RNN, mais aussi le besoin d'une optimisation plus fine de la régularisation. Par manque de temps nous laissons cela pour des travaux futurs.

Au-delà de tout cela, les résultats discutés ici sont très compétitifs par rapport aux meilleurs résultats de la littérature, comme nous le montrons dans la section suivante.

3.3.2. *Comparaison avec d'autres travaux*

Nous comparons ici les résultats obtenus par nos modèles avec ceux publiés dans la littérature. Pour que la comparaison soit équitable, nous comparons nos modèles *LD-RNN* utilisant la même information en entrée : les mots et les classes de mots.

Modèle	Mesure F1		
	Forward	*Backward*	Bidirectionnel
(Vukotic *et al.*, 2016) LSTM	95,12	–	95,23
(Vukotic *et al.*, 2016) GRU	**95,43**	–	**95,53**
(Dinarelli et Tellier, 2016a) E-RNN	94,73	93,61	94,71
(Dinarelli et Tellier, 2016a) J-RNN	94,94	94,80	94,89
(Dinarelli et Tellier, 2016a) I-RNN	95,21	94,64	94,75
LD-RNN Mots + Classes	95,31	95,42	**95,53**

Tableau 5. *Comparaison des résultats sur le corpus ATIS en termes de mesure* F1

Modèle	Mesure F1		
	Forward	*Backward*	Bidirectionnel
(Vukotic *et al.*, 2015) CRF		86,00	
(Vukotic *et al.*, 2015) E-RNN	81,94	–	–
(Vukotic *et al.*, 2015) J-RNN	83,25	–	–
(Vukotic *et al.*, 2016) LSTM	81,54	–	83,07
(Vukotic *et al.*, 2016) GRU	83,18	–	83,63
(Dinarelli et Tellier, 2016a) E-RNN	82,64	82,61	83,13
(Dinarelli et Tellier, 2016a) J-RNN	83,06	83,74	84,29
(Dinarelli et Tellier, 2016a) I-RNN	84,91	86,28	86,71
LD-RNN Mots + Classes	**85,46**	**86,59**	**87,16**

Tableau 6. *Comparaison des résultats sur le corpus MEDIA en termes de mesure* F1

Les résultats pour la tâche ATIS sont montrés dans le tableau 5. Ils sont comparés aux résultats de Vukotic *et al.* (2016) et Dinarelli et Tellier (2016a), ces derniers ayant été obtenus avec un modèle similaire à notre modèle *LD-RNN*.

Comme on peut le voir dans le tableau 5, tous les modèles obtiennent des très bons résultats sur cette tâche, au-delà de $94,5$ en $F1$, confirmant ce que nous avions anticipé dans la section 3.1 concernant la simplicité de cette tâche. Les modèles GRU et notre variante *LD-RNN* obtiennent des résultats équivalents $(95,53)$ et légèrement meilleurs par rapport aux autres, notamment avec leur version bidirectionnelle. Ceci est en soi un bon résultat, notre variante *LD-RNN* étant bien plus simple qu'un modèle GRU en termes de nombre de paramètres du modèle (voir la section 2.6). Les modèles comme LSTM et GRU sont réputés comme très efficaces pour modéliser le contexte côté source (les mots), ce qui constitue l'information la plus importante dans cette tâche : la meilleure taille du contexte de mots est de 11 (5 mots à gauche plus 5 à droite et le mot courant à étiqueter) dans nos modèles *LD-RNN*.

Si l'on compare ces résultats à ceux de Dinarelli et Tellier (2016a) avec un modèle de Jordan, qui utilise un contexte d'étiquettes de la même taille que nos modèles, nous pouvons conclure que l'avantage de notre variante *LD-RNN* vient de l'utilisation des plongements d'étiquettes et leur combinaison au niveau de la couche cachée.

Cette conclusion devient évidente quand on compare les résultats d'un RNN employant les plongements d'étiquettes aux autres RNN sur la tâche MEDIA. La comparaison des résultats avec la littérature pour cette tâche est donnée dans le tableau 6. Comme nous l'avons expliqué dans la section 3.1, cette tâche est bien plus difficile que

Modèle	CER
(Dinarelli *et al.*, 2011)	11.7
(Dinarelli et Rosset, 2011)	11.5
(Hahn *et al.*, 2010)	10.6
LD-RNN Mots	10.73 (10.63)
LD-RNN Mots + Classes	10.52 (10.15)
LD-RNN Mots + Classes + CC	**10.41** (**10.09**)
$LD\text{-}RNN_{LReLU}$ Mots + Classes + CC	**10.38** (10.16)

Tableau 7. *Résultats sur le corpus MEDIA comparés à d'autres travaux en termes de* Concept Error Rate *(CER) sur l'extraction des concepts seuls (sans les valeurs). Les valeurs entre parenthèses sont les meilleurs résultats des* 6 *moyennés pour chaque modèle*

la tâche ATIS, et ce pour plusieurs raisons, mais dans le contexte de cet article nous nous intéressons plus particulièrement aux dépendances entre étiquettes que nous prétendons modéliser de façon plus efficace avec les plongements d'étiquettes.

Dans ce contexte, nous notons qu'un modèle de Jordan classique, le modèle *J-RNN* de Dinarelli et Tellier (2016a), seul modèle traditionnel employant un contexte d'étiquettes, est plus efficace que les autres modèles classiques, y compris LSTM et GRU (F1 de $84,29$ avec J-RNN, contre $83,63$ pour le modèle GRU, deuxième meilleur modèle dans les RNN plus classiques). Nous notons également que, sur cette tâche, un CRF, modèle conçu pour l'étiquetage de séquences, est bien plus efficace que tous les RNN classiques (F1 $86,00$ avec le CRF de Vukotic *et al.* (2015)). Les seuls modèles capables d'obtenir des résultats meilleurs qu'un CRF sur cette tâche sont l'*I-RNN* de Dinarelli et Tellier (2016a) et notre variante *LD-RNN*, tous les deux employant les plongements d'étiquettes. Grâce à l'utilisation d'une couche cachée de type *ReLU* et d'une régularisation *dropout* sur la couche cachée et sur la couche des plongements, notre *LD-RNN* est le modèle le plus efficace aussi sur la tâche MEDIA.

Même si les résultats sur la tâche MEDIA commentés jusqu'ici sont très compétitifs, celle-ci est à la base conçue pour la reconnaissance automatique de la parole (De Mori *et al.*, 2008). Dans cette tâche, le but est d'extraire correctement les concepts que le système de dialogue va utiliser pour interpréter la requête de l'utilisateur. Bien que la mesure F1 y soit fortement corrélée, la mesure d'évaluation classique pour cette tâche est le *Concept Error Rate* (CER), défini de la même façon que le *Word Error Rate* pour la reconnaissance vocale, où les mots sont remplacés par les concepts.

Pour placer nos modèles sur une échelle absolue dans la tâche MEDIA, nous proposons une comparaison de nos meilleurs modèles avec ceux de la littérature en termes de CER, à savoir (Hahn *et al.*, 2010), (Dinarelli et Rosset, 2011) et (Dinarelli *et al.*, 2011). Cette comparaison est montrée dans le tableau 7. Les meilleurs modèles individuels publiés dans (Hahn *et al.*, 2010 ; Dinarelli et Rosset, 2011 ; Dinarelli *et al.*, 2011) sont des CRF, dont le CER est de $10,6$, $10,5$ et $11,7$, respectivement. Nous notons que ces modèles utilisent les mots et les classes comme entrées, mais aussi un certains nombre de traits classiques pour augmenter le pouvoir de généralisation du modèle sur les mots rares et/ou hors vocabulaire, tels que les préfixes et suffixes des mots, des traits binaires sur la capitalisation des mots, etc. Tous ces traits

fournissent au modèle le même type d'information qui peut être obtenu avec la convolution sur les caractères utilisée par nos modèles. Nous notons également que l'écart si important entre ces modèles CRF est dû au fait que le CRF de Hahn *et al.* (2010) est entraîné avec un critère d'apprentissage amélioré par rapport à un CRF classique, et fondé sur une notion de marge similaire à celle des *SVM* (Herbrich *et al.*, 2000 ; Hahn *et al.*, 2009). Par ailleurs, nous notons aussi que d'après les tests de significativité publiés dans (Dinarelli *et al.*, 2011), une différence de $0,1$ dans le CER est déjà significative sur MEDIA.

Notre meilleur modèle *LD-RNN* obtient un CER de $10,41$. À notre connaissance, il s'agit du meilleur résultat obtenu sur cette tâche avec un modèle individuel. En effet les auteurs de (Hahn *et al.*, 2010) obtiennent un CER de $10,2$ avec un modèle *ROVER* (Fiscus, 1997) combinant 6 modèles individuels. Comme cela a déjà été fait dans d'autres travaux, par exemple (Yao *et al.*, 2013), plutôt que moyenner 6 résultats obtenus avec autant de modèles, il est possible de lancer toujours 6 expériences différentes et de ne garder que le meilleur modèle des 6 sur les données de développement d'une tâche donnée.[10] Les résultats obtenus avec nos modèles *LD-RNN* en suivant cette procédure sont donnés dans le tableau 7 entre parenthèses. Notre meilleur résultat est un CER de $10,09$, constituant le meilleur résultat absolu jusqu'ici sur la tâche MEDIA.

3.4. *Analyses*

Pour montrer que les meilleurs résultats de notre variante de RNN sur MEDIA sont dus à une meilleure modélisation des séquences, nous avons fait des analyses des résultats en les comparant avec ceux obtenus par des réseaux d'Elman et de Jordan entraînés avec le code mis à disposition par Mesnil *et al.* (2013).[11]

La plus grosse différence dans les résultats est que le modèle d'Elman commence l'annotation d'un concept avec l'étiquette *I* (schéma d'annotation *BIO*). Sans être une erreur grave en soi, c'est une conséquence claire d'une décision locale. Le modèle de Jordan souffre moins de ce problème puisqu'il intègre les étiquettes dans son contexte. Les plongements d'étiquettes utilisés par la variante *LD-RNN* rendent ce modèle bien plus robuste, ce dernier ne produisant aucune erreur de segmentation (*BIO*). Ceci est d'autant plus remarquable que nous n'avons jamais observé une telle précision même sur des modèles CRF optimisés pour la tâche MEDIA.

Le tableau 8 montre les 5 erreurs d'insertion les plus fréquentes d'un RNN de Jordan et de *LD-RNN*. Le modèle de Jordan fait plus d'erreurs sur des concepts difficiles à annoter pour un modèle qui n'arrive pas à prendre en compte les dépendances entre étiquettes. Un exemple frappant est le concept *command-tache*, utilisé pour désigner de façon générale la requête de l'utilisateur. Ce concept est instancié par des séquences de mots relativement longues. Le modèle de Jordan a tendance à le segmenter en plu-

10. Éventuellement en sauvegardant la graine utilisé pour initialiser les poids du modèle de façon à ce que l'entraînement puisse être reproduit.
11. http ://deeplearning.net/tutorial/rnnslu.html

Jordan RNN	LD-RNN
1 : 46 -> connectprop	1 : 39 -> lienref-coref
2 : 44 -> lienref-coref	2 : 36 -> connectprop
3 : 28 -> command-tache	3 : 25 -> nombre
4 : 26 -> nombre	4 : 11 -> reponse
5 : 13 -> reponse	5 : 11 -> loc.-distancerelative

Tableau 8. *Les* 5 *erreurs les plus fréquentes sur MEDIA avec un RNN de Jordan et LD-RNN*

	Jordan RNN			LD-RNN		
label	P	R	F1	P	R	F1
connectprop	74,03	76,05	75,03	82,06	70,98	76,12
lienRef-coRef	82,56	84,87	83,70	88,89	87,91	88,40
command-tache	73,28	77,31	75,24	82,89	80,47	81,67

Tableau 9. *Détails sur les concepts du corpus MEDIA les plus difficiles à annoter correctement*

sieurs concepts, en introduisant une étiquette vide (*O*). Ce même comportement est observé avec un modèle d'Elman. Puisqu'il ne prend pas en compte les étiquettes dans son contexte, nous pensons qu'il sera observé aussi dans un LSTM. En revanche, ce concept ne fait pas partie des 5 erreurs les plus fréquentes du *LD-RNN*.

Les concepts *connectprop* et *lienref-coref* sont utilisés pour les coréférences. Ceux-ci sont de loin les concepts les plus difficiles à annoter dans la tâche MEDIA, leur annotation dépendant fortement du contexte. Cela est visible dans le tableau 8, où ils sont les concepts les plus sujets à erreur. Sur ces concepts aussi, le modèle *LD-RNN* obtient des résultats bien meilleurs, comme nous le montrons dans le tableau 9.

4. Conclusion

Dans cet article nous avons proposé une variante de RNN employant des plongements d'étiquettes et utilisant un large contexte d'étiquettes comme information additionnelle pour prédire la prochaine étiquette dans une séquence. Nous avons motivé ce type d'architecture comme étant plus efficace que les autres RNN pour modéliser des dépendances entre étiquettes. Les résultats sur deux tâches de compréhension automatique de la parole montrent que i) sur une tâche relativement simple comme ATIS, notre variante obtient les mêmes résultats que des réseaux plus complexes et réputés plus efficaces comme LSTM et GRU ; ii) sur la tâche MEDIA, dans laquelle la modélisation des dépendances entre étiquettes est cruciale, notre variante obtient des résultats largement meilleurs que tous les autres RNN. Comparée aux meilleurs modèles publiés dans la littérature en termes de *Concept Error Rate*, notre variante s'avère plus performante, obtenant un CER à l'état de l'art de $10,09$.

Remerciements

Ce travail a été financé en partie par le projet ANR Democrat ANR-15-CE38-0008.

5. Bibliographie

Bengio Y., « Practical recommendations for gradient-based training of deep architectures », *CoRR*, 2012.

Bengio Y., Ducharme R., Vincent P., Jauvin C., « A Neural Probabilistic Language Model », *JOURNAL OF MACHINE LEARNING RESEARCH*, vol. 3, p. 1137-1155, 2003.

Bengio Y., Simard P., Frasconi P., « Learning Long-term Dependencies with Gradient Descent is Difficult », *Trans. Neur. Netw.*, vol. 5, n° 2, p. 157-166, March, 1994.

Bonneau-Maynard H., Ayache C., Bechet F., Denis A., Kuhn A., Lefèvre F., Mostefa D., Qugnard M., Rosset S., Servan S. Vilaneau J., « Results of the French Evalda-Media evaluation campaign for literal understanding », *LREC*, Genoa, Italy, p. 2054-2059, May, 2006.

Chen D., Manning C., « A Fast and Accurate Dependency Parser using Neural Networks », *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, Doha, Qatar, p. 740-750, October, 2014.

Chiu J. P. C., Nichols E., « Named Entity Recognition with Bidirectional LSTM-CNNs », *CoRR*, 2015.

Cho K., van Merrienboer B., Gülçehre Ç., Bougares F., Schwenk H., Bengio Y., « Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation », *CoRR*, 2014.

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

Collobert R., Weston J., Bottou L., Karlen M., Kavukcuoglu K., Kuksa P., « Natural Language Processing (Almost) from Scratch », *J. Mach. Learn. Res.*, vol. 12, p. 2493-2537, November, 2011.

Dahl D. A., Bates M., Brown M., Fisher W., Hunicke-Smith K., Pallett D., Pao C., Rudnicky A., Shriberg E., « Expanding the Scope of the ATIS Task : The ATIS-3 Corpus », *Proceedings of the Workshop on Human Language Technology*, HLT '94, Association for Computational Linguistics, Stroudsburg, PA, USA, p. 43-48, 1994.

De Mori R., Bechet F., Hakkani-Tur D., McTear M., Riccardi G., Tur G., « Spoken Language Understanding : A Survey », *IEEE Signal Processing Magazine*, vol. 25, p. 50-58, 2008.

Dinarelli M., Moschitti A., Riccardi G., « Discriminative Reranking for Spoken Language Understanding », *IEEE Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 20, p. 526-539, 2011.

Dinarelli M., Rosset S., « Hypotheses Selection Criteria in a Reranking Framework for Spoken Language Understanding », *Conference of Empirical Methods for Natural Language Processing*, Edinburgh, U.K., p. 1104-1115, Jul, 2011.

Dinarelli M., Tellier I., « Etude des reseaux de neurones recurrents pour etiquetage de sequences », *Actes de la 23eme conférence sur le Traitement Automatique des Langues Na-*

turelles, Association pour le Traitement Automatique des Langues, Paris, France, Juillet, 2016a.

Dinarelli M., Tellier I., « Improving Recurrent Neural Networks For Sequence Labelling », *CoRR*, 2016b.

Dinarelli M., Tellier I., « New Recurrent Neural Network Variants for Sequence Labeling », *Proceedings of the 17th International Conference on Intelligent Text Processing and Computational Linguistics*, Lecture Notes in Computer Science (Springer), Konya, Turkey, Avril, 2016c.

Dupont Y., Dinarelli M., Tellier I., « Label-Dependencies Aware Recurrent Neural Networks », *Proceedings of the 18th International Conference on Computational Linguistics and Intelligent Text Processing*, Lecture Notes in Computer Science (Springer), Budapest, Hungary, April, 2017.

Elman J. L., « Finding structure in time », *COGNITIVE SCIENCE*, vol. 14, nº 2, p. 179-211, 1990.

Fiscus J. G., « A Post-Processing System to Yield Reduced Word Error Rates : Recogniser Output Voting Error Reduction (ROVER) », *Proceedings 1997 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, Santa Barbara, CA, p. 347-352, December, 1997.

Hahn S., Dinarelli M., Raymond C., Lefèvre F., Lehen P., De Mori R., Moschitti A., Ney H., Riccardi G., « Comparing Stochastic Approaches to Spoken Language Understanding in Multiple Languages », *IEEE Transactions on Audio, Speech and Language Processing (TASLP)*, 2010.

Hahn S., Lehnen P., Heigold G., Ney H., « Optimizing CRFs For SLU Tasks In Various Languages Using Modified Training Criteria », *Proceedings of the International Conference of the Speech Communication Assosiation (Interspeech)*, Brighton, U.K., 2009.

He K., Zhang X., Ren S., Sun J., « Delving Deep into Rectifiers : Surpassing Human-Level Performance on ImageNet Classification », *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, p. 1026-1034, 2015.

Herbrich R., Graepel T., Obermayer K., *Large Margin Rank Boundaries for Ordinal Regression*, MIT Press, March, 2000.

Hochreiter S., Bengio Y., Frasconi P., Schmidhuber J., « Gradient flow in recurrent nets : the difficulty of learning long-term », *in* Kremer, Kolen (eds), *A Field Guide to Dynamical Recurrent Neural Networks*, IEEE Press, 2001.

Hochreiter S., Schmidhuber J., « Long Short-Term Memory », *Neural Comput.*, vol. 9, nº 8, p. 1735-1780, November, 1997.

Huang Z., Xu W., Yu K., « Bidirectional LSTM-CRF models for sequence tagging », *arXiv preprint arXiv :1508.01991*, 2015.

Jordan M. I., « Serial Order : A Parallel, Distributed Processing Approach », *in* J. L. Elman, D. E. Rumelhart (eds), *Advances in Connectionist Theory : Speech*, Erlbaum, Hillsdale, NJ, 1989.

Lample G., Ballesteros M., Subramanian S., Kawakami K., Dyer C., « Neural architectures for named entity recognition », *arXiv preprint arXiv :1603.01360*, 2016.

Ma X., Hovy E., « End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF », *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016*, 2016.

Mesnil G., He X., Deng L., Bengio Y., « Investigation of Recurrent-Neural-Network Architectures and Learning Methods for Spoken Language Understanding », *Interspeech 2013*, August, 2013.

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

Mikolov T., Karafiát M., Burget L., Cernocký J., Khudanpur S., « Recurrent neural network based language model », *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, p. 1045-1048, 2010.

Mikolov T., Kombrink S., Burget L., Cernocky J., Khudanpur S., « Extensions of recurrent neural network language model. », *ICASSP*, IEEE, p. 5528-5531, 2011.

Mikolov T., Yih W., Zweig G., « Linguistic Regularities in Continuous Space Word Representations », *Human Language Technologies : Conference of the North American Chapter of the Association of Computational Linguistics*, p. 746-751, 2013b.

Pennington J., Socher R., Manning C. D., « GloVe : Global Vectors for Word Representation », *Empirical Methods in Natural Language Processing (EMNLP)*, p. 1532-1543, 2014.

Ramshaw L., Marcus M., « Text chunking using transformation-based learning », *Proceedings of the 3rd Workshop on Very Large Corpora*, Cambridge, MA, USA, p. 84-94, June, 1995.

Raymond C., Riccardi G., « Generative and Discriminative Algorithms for Spoken Language Understanding », *Proceedings of the International Conference of the Speech Communication Assosiation (Interspeech)*, Antwerp, Belgium, p. 1605-1608, August, 2007.

Schuster M., Paliwal K., « Bidirectional Recurrent Neural Networks », *Trans. Sig. Proc.*, vol. 45, n° 11, p. 2673-2681, nov, 1997.

Srivastava N., Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R., « Dropout : A Simple Way to Prevent Neural Networks from Overfitting », *Journal of Machine Learning Research*, vol. 15, p. 1929-1958, 2014.

Vukotic V., Raymond C., Gravier G., « Is it time to switch to Word Embedding and Recurrent Neural Networks for Spoken Language Understanding ? », *InterSpeech*, Dresde, Germany, September, 2015.

Vukotic V., Raymond C., Gravier G., « A step beyond local observations with a dialog aware bidirectional GRU network for Spoken Language Understanding », *Interspeech*, San Francisco, United States, September, 2016.

Werbos P., « Backpropagation through time : what does it do and how to do it », *Proceedings of IEEE*, vol. 78, p. 1550-1560, 1990.

Yao K., Zweig G., Hwang M.-Y., Shi Y., Yu D., « Recurrent Neural Networks for Language Understanding », Interspeech, August, 2013.