Réseau de neurones profond pour l'étiquetage morpho-syntaxique

Jérémie Tafforeau[1]
(1) LIF, AMU, CNRS, 163 avenue de Luminy, 13288 Marseille Cedex 9
jeremie.tafforeau@lif.amu-mrs.fr

Résumé. L'analyse syntaxique et sémantique de langages non-canoniques est principalement limitée par le manque de corpus annotés. Il est donc primordial de mettre au point des systèmes robustes capables d'allier références canoniques et non-canoniques. Les méthodes exploitant la théorie des réseaux de neurones profonds ont prouvé leur efficacité dans des domaines tels que l'imagerie ou les traitements acoustiques. Nous proposons une architecture de réseau de neurones appliquée au traitement automatique des langages naturels, et plus particulièrement à l'étiquetage morpho-syntaxique. De plus, plutôt que d'extraire des représentations empiriques d'une phrase pour les injecter dans un algorithme de classification, nous nous inspirons de récents travaux portant sur l'extraction automatique de représentations vectorielles des mots à partir de corpus non-annotés. Nous souhaitons ainsi tirer profit des propriétés de linéarité et de compositionnalité de tels plongements afin d'améliorer les performances de notre système.

Abstract. Syntactic and semantic parsing of non-canonical languages is mainly restricted by the lack of labelled data sets. It is thus essential to develop strong systems capable of combining canonical and non-canonical text corpora. Deep Learning methods proved their efficiency in domains such as imaging or acoustic process. We propose neural network architecture applied to natural languages processing. Furthermore, instead of extracting from the sentence a rich set of hand-crafted features wich are the fed to a standard classification algorithm, we drew our inspiration from recent papers about the automatic extraction of word embeddings from large unlabelled data sets. On such embeddings, we expect to benefit from linearity and compositionality properties to improve our system performances.

Mots-clés : TALN, Étiquetage morpho-syntaxique, Apprentissage Automatique, Réseau de Neurones Profond, Plongements.

Keywords: NLP, Part-of-Speech Tagging, Machine Learning, Deep Neural Network, Embeddings.

1 Introduction

Dans le cadre du projet européen SENSEI fondé sur l'étude des conversations humaines, nous allons être amené à analyser syntaxiquement et sémantiquement des corpus de langages non-canoniques tels que des transcriptions de conversations téléphoniques ou des commentaires web. Le manque de corpus annotés dans ces domaines force l'utilisation de corpus alliant références canoniques et non-canoniques. Les performances des approches statistiques sont particulièrement affectées par un aussi radical changement de contexte (Giesbrecht & Evert, 2009). Nous nous proposons de mettre au point un réseau de neurones profond égalant les résultats de références sur des données journalistiques avant de considérer, dans de futurs travaux, son adaptation aux corpus non-canoniques.

Les méthodes exploitant la théorie de l'apprentissage automatique de réseaux de neurones profonds ou *Deep Learning* ont prouvé leur robustesse sur des tâches complexes du domaines de l'imagerie et du traitement acoustique. Nous nous situons dans la lignée de récents travaux misant sur l'auto-adaptabilité de tels systèmes dans le cas de traitement d'informations partielles et bruitées (Collobert *et al.*, 2011). Nous proposons une architecture de réseau de neurones profond appliquée aux traitements automatiques des langages naturels. Afin d'éprouver notre méthodologie, nous nous fixons comme contexte expérimental l'étiquetage morpho-syntaxique ou *POS tagging* du corpus Penn TreeBank. Il s'agit en effet d'une tâche bien connue du traitement automatique de la langue (Manning, 2011) pour laquelle existent de nombreux résultats de référence.

Après quelques rappels généraux sur la théorie des réseaux de neurones, nous exposerons de récents travaux portant sur l'extraction de représentations vectorielles des mots d'un texte (Mikolov *et al.*, 2013a). Ces plongements ou *embeddings* présentent d'intéressantes propriétés de linéarité et de compositionnalité (Mikolov *et al.*, 2013b) que nous souhaitons capturer par l'apprentissage au plus profond du réseau de neurones. En résulte un gain en terme de rapidité de convergence ainsi qu'une approche plus fine des mots inconnus.

2 Réseaux de Neurones

L'approche conventionnelle afin d'étiqueter une séquence de mots est la suivante : extraire des représentations de la phrase en cours de traitement et les injecter dans un algorithme de classification, par exemple une machine à vecteurs de support (*SVM*), bien souvent à l'aide d'un *kernel trick*. Nous envisageons ici une approche radicalement différente : nos données seront aussi peu pré-traitées que possible afin de permettre à notre réseau de neurones d'en extraire, couche après couche, ses propres représentations internes qui seront entrainées par *rétro-propagation* afin de gagner en pertinence tout au long de l'apprentissage.

2.1 Architecture & Notations

Il existe un grand nombre d'architectures possibles lorsqu'il s'agit d'assembler des neurones en réseau. On pourra citer les *perceptrons multi-couches*, les *réseaux de neurones convolutionels* (Collobert *et al.*, 2011), les *machines de Boltzmann restreintes* (Fischer & Igel, 2012) ou encore d'autres types de *réseaux de neurones récurrents* (Toutanova *et al.*, 2003). Si les propriétés de tels réseaux sont connues depuis des décennies, les architectures comprenant plus de trois couches de neurones ne présentaient pas les améliorations attendues. C'est désormais le cas grâce à de récentes évolutions matérielles et algorithmiques. On parle depuis de réseaux de neurones profonds ou *deep neural networks*.

Considérons un réseau de neurones $f_\theta()$ de paramètres θ. Tout réseau de neurones en aval ou *feed-forward neural networks* à L couches peut être interprété comme la composition des fonctions $f^l_\theta()$ associées à chaque couche l :

$$f_\theta() = f^L_\theta(f^{L-1}_\theta(...f^1_\theta()...)) = f^L_\theta \circ f^{L-1}_\theta \circ ... \circ f^1_\theta()$$

En cela, un réseau de neurones peut être considéré comme un système d'approximation fonctionnelle. Dans toute la suite, nous noterons $(|input| \times |output|)$ afin de caractériser une couche neuronale ayant $|input|$ neurones d'entrée et $|output|$ neurones de sortie. Ce même formalisme permet de décrire des réseaux aux structures neuronales plus complexes, cf. FIGURE 1.

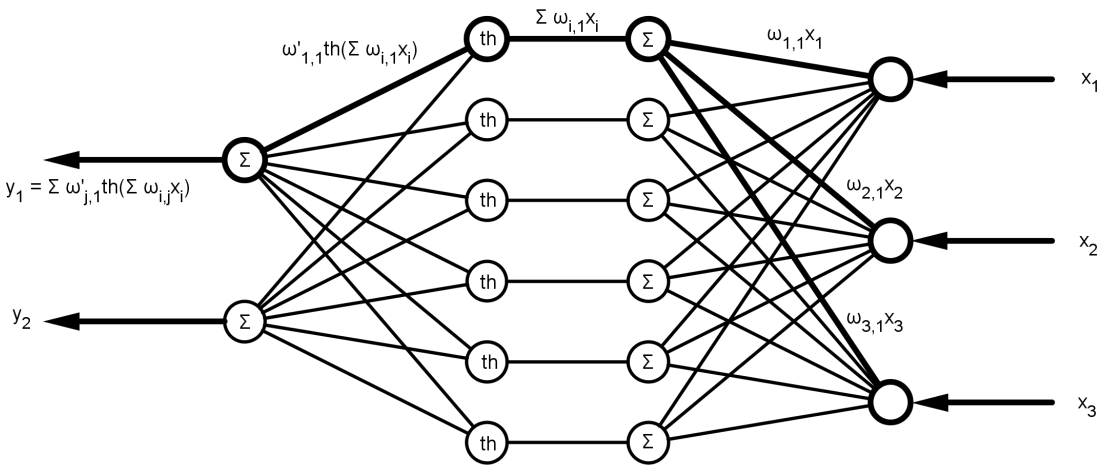

FIGURE 1 – Exemple de réseau de neurones ($3 \times 6 \times 2$)

Chaque couche neuronale réalise un plongement entre des espaces de dimensions variables en associant un vecteur de sortie à chaque vecteur d'entrée. Ainsi, compressions et dilatations des représentations internes du réseau se composent afin d'approximer au mieux une fonction pouvant permettre la classification de nos données d'apprentissage.

2.2 Structure Neuronale

Nous considérons par la suite une structure classique de réseau neuronal basée sur celle du *perceptron multi-couches* ou *multi-layer perceptron*. Elle se compose de couches linéaires réalisant une transformation affine de leur vecteur d'entrée, et de couches non-linéaires, ayant une tangente hyperbolique comme fonction d'activation, cf FIGURE 2. On alterne couches linéaires et non-linéaires afin de pouvoir approximer toute fonction, même hautement non-linéaire. En effet, si aucune non-linéarité n'est introduite dans le réseau, alors ce dernier se résumerait en un simple modèle linéaire. Enfin, la dernière couche de notre réseau compte autant de neurones que le problème considéré ne compte de classes. Chaque sortie peut ainsi être interprétée comme le score de la classe correspondante pour un exemple donnée en entrée du réseau.

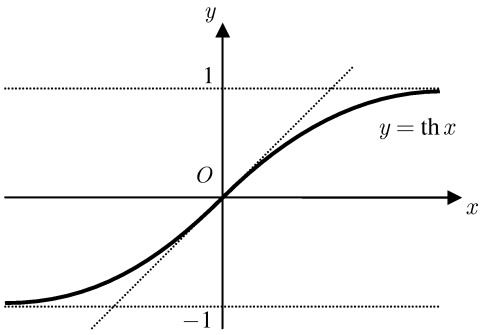

FIGURE 2 – tangente hyperbolique

2.3 Apprentissage

Tous nos réseaux ont été entrainés par maximum de vraisemblance sur l'ensemble d'apprentissage par descente de gradient stochastique. Si on note θ l'ensemble des paramètres du réseau, entrainé sur l'ensemble d'apprentissage $\mathcal{T}$, on souhaite maximiser la log-vraisemblance :

$$\theta \mapsto \sum_{(x,y)\in\mathcal{T}} log\ p(y|x,\theta),$$

où x correspond à un vecteur de *features* et y représente l'étiquette correspondante. Soit un exemple x donné en entrée, on note $[f_\theta(x)]_i$ le score associé à la i$^{\text{ième}}$ étiquette par notre réseau de paramètre θ. La probabilité $p()$ est calculé à partir des sorties du réseau de neurones au moyen d'un *softmax* :

$$p(i|x,\theta) \ = \ \frac{e^{[f_\theta(x)]_i}}{\sum_j e^{[f_\theta(x)]_j}}$$

Ce qui nous permet d'exprimer aisément la log-vraisemblance :

$$log\ p(y|x,\theta) \ = \ [f_\theta(x)]_y - log(\sum_j e^{[f_\theta(x)]_j})$$

Maximiser cette log-vraisemblance au moyen d'un gradient stochastique s'effectue en sélectionnant aléatoirement un exemple d'apprentissage (x, y) et en réalisant une descente de gradient :

$$\theta \leftarrow \theta \ + \ \lambda \frac{\delta\ log\ p(y|x,\theta)}{\delta\theta},$$

où λ est le taux d'apprentissage ou *learning rate* considéré. Cela revient à explorer l'espace des fonctions approximables par notre réseau en minimisant pas à pas l'erreur de classification.

3 Représentations vectorielles du texte pour le *POS tagging*

3.1 Le *POS tagging*, un problème de classification

Notre contexte expérimental est le *POS tagging* qui a pour but d'étiqueter chaque mots en fonction de son rôle syntaxique, par exemple, nom, adjectif, adverbe, etc. Les sections 0-18 ainsi que 19-21 du Penn TreeBank ont été respectivement utilisées comme ensemble d'apprentissage et de développement, tandis que les sections 22-24 furent réservées à la phase de test. Afin de minimiser le nombre de mots composant notre vocabulaire, nous avons modifié notre corpus afin qu'il ne comporte que des mots en minuscules. De même, toutes les séquences de chiffres formant un nombre entier ont été remplacées par la chaîne _NUMBER_ (*ex. 1 ; 1,000 ; 1,000,000*) et celles formant un nombre réel par la chaîne _REAL_ (*ex. 3.14 ; 6,378.137*).

Dataset	#phrases	#mots	#mots inconnus	#classes
Training	38.219	912.344	0	48
Development	5.527	131.768	4.467	48
Test	5.462	129.654	3.649	48

TABLE 1 – Caractéristiques du corpus Penn TreeBank

Les meilleurs systèmes pour cette tâche sont généralement entraînés sur une fenêtre de texte et servent de fondations à des algorithmes bidirectionnels apportant une information de séquence, cf.*HMM & Viterbi*. Parmis les *features* récurrentes, on trouve les *tags* contextuels i.e les étiquettes des mots suivant ou précédant le mot à étiqueter, les *n-grams* ainsi que des *features* empiriques ou linguistiquement motivées, dédiées à la gestion des mots inconnus. Comme dans la plupart des travaux nous ayant précédés, nous abordons le *POS Tagging* comme un problème de classification. Mettant de côté l'aspect séquentiel, nous souhaitons attribuer indépendamment à chaque mot d'une phrase son étiquette morpho-syntaxique correspondante. Il nous est donc nécessaire de fixer une représentation vectorielle propre à chaque mot qui constitueront les entrées de notre classifieur.

Model	All Words Accuracy	Unknown Words Accuracy
Hidden Markov model (Brants, 2000)	96,46%	85,86%
Support machine vectors (Giménez & Márquez, 2004)	97,16%	89,01%
Maximum entropy cyclic dependency network (Manning, 2011)	97,32%	90,79%

TABLE 2 – Résultats de référence pour le *POS tagging* sur le Penn TreeBank

3.2 Représentations en vecteur d'index lexical

Par soucis d'efficacité, les mots sont fournies au réseau sous forme d'indices tirés d'un lexique. C'est pourquoi la première étape de notre modèle est d'établir un lien entre indices et représentations vectorielles au moyen d'une table d'association. Soit $|V|$ la taille de notre vocabulaire. Nous considérons dans un premier temps comme représentation du mot w d'indice lexical n_w le vecteur *"creux"* :

$$W = \begin{cases} W_{n_w} & = & 1 \\ W_i & = & 0 \quad \forall i \neq n_w \end{cases}$$

Ces représentations vectorielles doivent par la suite être combinées afin d'étiqueter un par un chaque mot composant la phrase en cours de traitement. Prenons comme hypothèse que seuls les mots voisins du mot considéré influent sur son étiquetage. Soit un mot w a étiqueter, on considère une fenêtre de taille fixe d_{win} centrée sur w. En associant à chaque mot composant cette fenêtre sa représentation vectorielle, nous obtenons une matrice de taille $d_{win} \times |V|$, cf FIGURE 3. Cette matrice peut être vue comme un vecteur de dimension $d_{win}.|V|$ par concaténation des colonnes, qui peut être injecté en entrée du réseau de neurones.

w_1	w_2	w_3	
0	0	0	0
.	.	.	
.	.	.	
1	0	0	n_{w_1}
.	.	.	
.	.	.	
0	0	1	n_{w_3}
.	.	.	
.	.	.	
0	1	0	n_{w_2}
.	.	.	
.	.	.	
0	0	0	$\|V\|$

FIGURE 3 – Représentation vectorielle associée à un triplet de mot (w_1, w_2, w_3)

3.3 Extraction automatique de représentations vectorielles

Il est possible d'extraire à partir de données non-annotées des plongements, ou *embeddings*, de plus petite dimension afin d'obtenir de meilleurs représentations vectorielles de nos mots. Dans nos expériences, nous nous basons sur l'approche WORD2VEC (Mikolov *et al.*, 2013a). Cette dernière consiste à entraîner un réseau de neurones linéaire i.e sans couche cachée non-linéaire. La matrice des poids de la couche linéaire peut ainsi être interprétée comme une projection linéaire, permettant de passer de l'espace des mots à un espace de dimension réduite. Deux heuristiques d'apprentissage ont été proposées afin de calibrer ce plongement, cf. FIGURE 4.

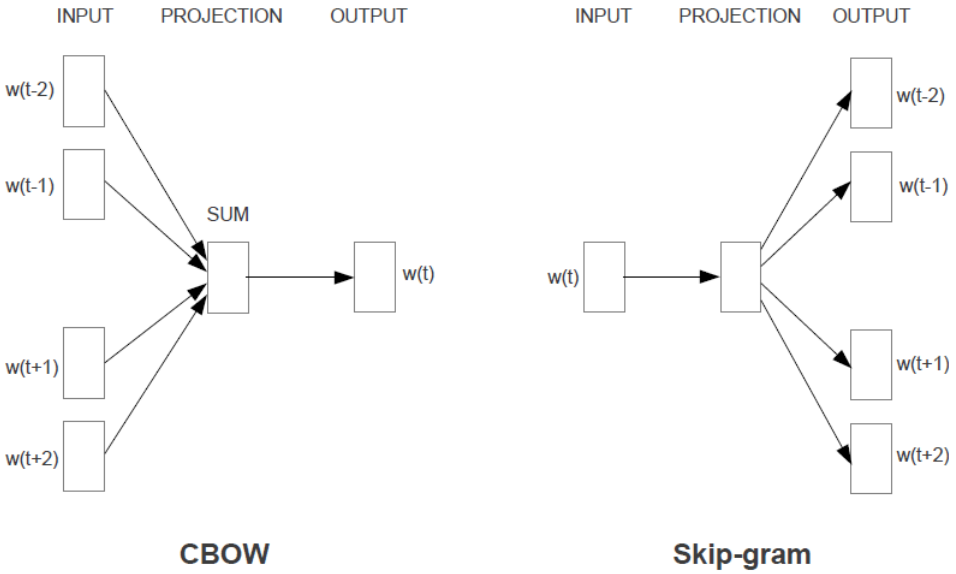

FIGURE 4 – WORD2VEC Architectures

L'approche *Continuous Bag of Words* (CBOW) consiste à entraîner le réseau à prédire un mot à partir de son contexte. À l'inverse, l'approche *Skip-gram* se propose de prédire les mots encadrant un mot donné en entrée. Une fois l'un de ses deux entraînements arrivé à son terme, les plongements sont extraits des poids de la couche linéaire du réseau. La dimensions n de ces plongements est ainsi librement paramétrable car totalement déterminée par le nombre de neurones composant cette couche de projection. Il s'agit donc d'un hyper-paramètre de notre système.

Ainsi, deux mots proches dans ce nouvel espace de représentations sont syntaxiquement et/ou sémantiquement proches car c'est la promiscuité de leur contexte respectif qui les a rapprochés dans ce nouvel espace. Il est bon de noter que ces dernières présentent également des propriétés de linéarité et de compositionalité (Mikolov *et al.*, 2013b) alors qu'aucune contrainte ne fut fixée en ce sens au cours de l'apprentissage. Notre architecture est capable de tirer partie de telles *features*.

4 Architectures proposées

L'ensemble des architectures réseaux proposées ont été réalisées à l'aide de la librairie *Torch7* car en plus de proposer une implémentation intuitive, elle a montré de bonnes performances dans les tests de (Collobert *et al.*, 2012).

4.1 Perceptron Multi-Couche Naïf

Il s'agit d'une implémentation standard d'un *Multi-Layer Perceptron* (*MLP*). Les entrées de ce réseau sont les concaténations de vecteurs *"creux"* d'index lexical. La structure de la couche cachée non-linéaire est un hyper-paramètre du réseau. Dans la suite et par soucis de clarté, nous considèrerons qu'elle ne se compose que d'une seule couche non-linéaire constituée de $d_{win} * n$ neurones tandis qu'elle sera bien plus complexe lors de l'exposé de nos expérimentations. En vu de permettre au réseau d'appréhender les mots de l'ensemble de test absents de l'ensemble d'apprentissage (en d'autres termes les mots inconnus), nous avons considéré les mots n'ayant qu'une seule occurrence au sein de notre corpus d'apprentissage comme inconnus. Ces derniers ont ainsi été remplacés dans notre corpus par la chaîne _UNK_.

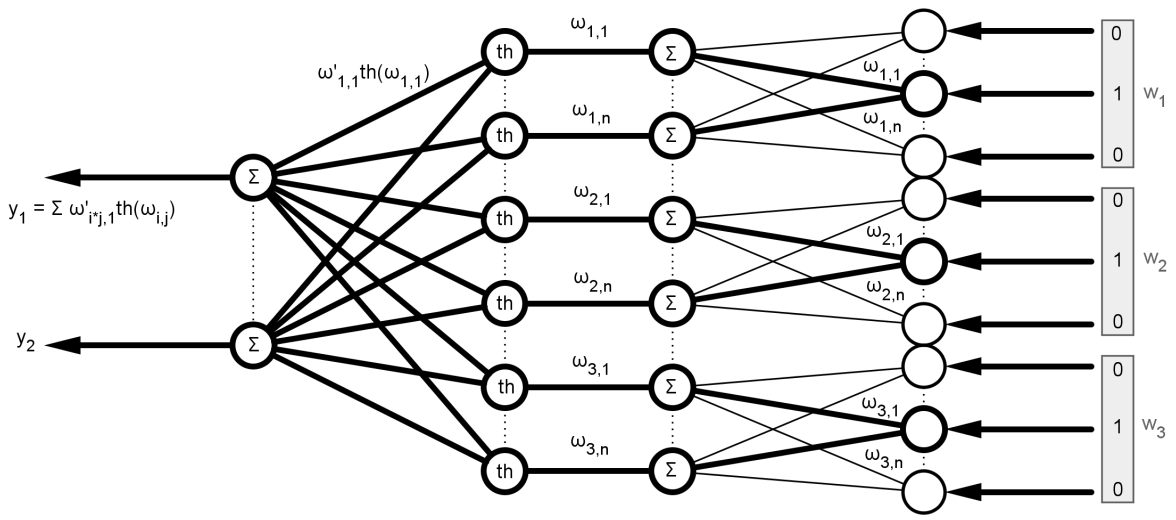

FIGURE 5 – Exemple de *MLP* à une couche cachée dans le cas d'une classification à 2 classes et d'une fenêtre de taille 3. On applique un *softmax* à nos sorties afin d'obtenir des scores assimilables à des probabilités . Les neurones $\sum$ réalisent une combinaison linéaire pondérée de leurs entrées tandis que les neurones *th* forment notre couche non-linéaire appliquant une tangente hyperbolique. On a représenté en gras les connexions réellement sollicitées par l'exemple d'apprentissage considéré.

On remarque immédiatement que la taille de la couche d'entrée, constituée de $d_{win} * |V|$ neurones, est directement proportionnelle au nombre de mots composant notre vocabulaire. Pourtant, pour un exemple d'apprentissage donné, très peu de connexions sont réellement sollicitées au sein de cette couche puisque seules d_{win} dimensions d'entrée sont effectivement actives i.e de valeur non-nulle. Dans la pratique, cette implémentation n'est pas efficace du fait du trop grand nombre de connexions qui la composent.

4.2 Initialisations des poids de la couche d'entrée par nos plongements

Nous souhaitons initialiser les poids de notre couche d'entrée à l'aide des plongements extraits, grâce à l'approche CBOW de WORD2VEC, d'un important corpus non-annoté tiré de *Wikipédia*, cf TABLE 3. Nous espérons obtenir ainsi une meilleur

convergence de notre modèle. De plus, nous nous proposons de réaliser une couche qui minimise le coût d'une descente de gradient dans le cas de vecteurs d'entrée *"creux"* afin de diminuer la complexité de la phase d'apprentissage. On expose pour ce faire deux implémentations distinctes.

#phrases	#mots	$\|V\|$
84.805.805	1.709.107.335	1.556.817

TABLE 3 – Caractéristiques du corpus d'apprentissage de WORD2VEC

4.2.1 Initialisation Statique (*SMLP*)

Nous considérons dans un premier temps un réseau simulant une couche d'entrée statique dont les poids ont été initialisés par nos plongements, cf. FIGURE 6. La couche d'entrée initiale y a été remplacée par un pré-traitement des entrées qui sont désormais les concaténations des plongements associés à chaque mot composant la fenêtre initialement injectée en entrée. Ce réseau simule un *perceptron multi-couches* à la différence que la *rétro-propagation* n'atteint pas les poids de la couche d'entrée qui restent par conséquent constants tout au long de l'apprentissage.

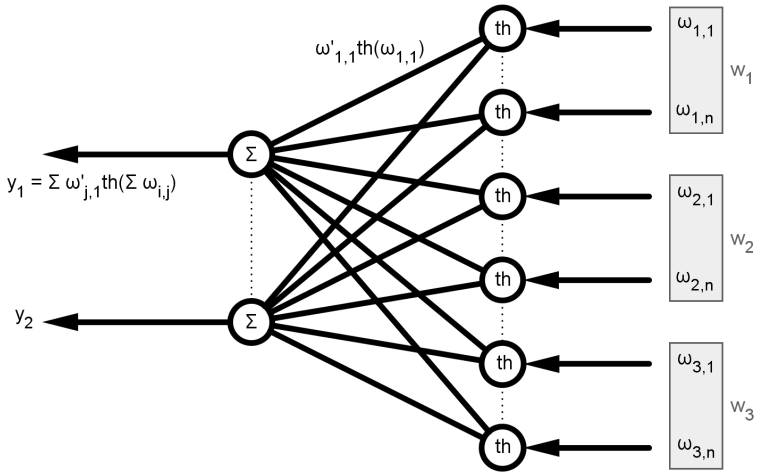

FIGURE 6 – Architecture *SMLP* équivalente au *MLP* décrit en FIGURE 5. Chaque mot $w_{i\in[1..3]}$ composant notre exemple d'apprentissage est désormais représenté comme un vecteur ω_i de dimension n. C'est la concaténation de ces plongements qu'on injecte en entrée du réseau. On obtient ainsi un réseau au comportement analogue au précédent, s'abstrayant des connexions inactives au prix du gel des paramètres de la couche d'entrée initiale.

4.2.2 Initialisation Dynamique (*DMLP*)

Nous considérons désormais un réseau permettant à la *rétro-propagation* de raffiner nos plongements, cf. FIGURE 7. À chaque mot w de notre dictionnaire, nous associons, dans une bibliothèque L de couches linéaires, une couche L_w de structure $(1 \times n)$ dont les n poids représentent nos plongements. Lorsqu'on souhaite injecter une fenêtre de mots (w_1,w_2,w_3) dans notre réseau, on remplace la couche d'entrée par la concaténation des couches linéaires associées L_{w_1}, L_{w_2} et L_{w_3}. La descente de gradient stochastique se propage ainsi jusque dans notre nouvelle couche d'entrée, modifiant par la même les poids des couches de notre bibliothèque et donc nos plongements.

Les entrées sont forcées à 1 et représentent les dimensions actives des vecteurs d'index lexical initiaux. De cette manière, la couche d'entrée simule les connexions sollicitées pour un mot donné et s'abstrait ainsi de toutes celles inactives dans le cas de vecteur *"creux"*. Cette implémentation est réalisée à l'aide de partages dynamiques de paramètres afin de manipuler nos couches d'entrées interchangeables. Cette architecture permet un raffinage des plongements initiaux tout au long de l'apprentissage en fonction de la tâche considérée tout en minimisant la taille de la structure neuronale nécessaire et donc le coût d'une descente de gradient.

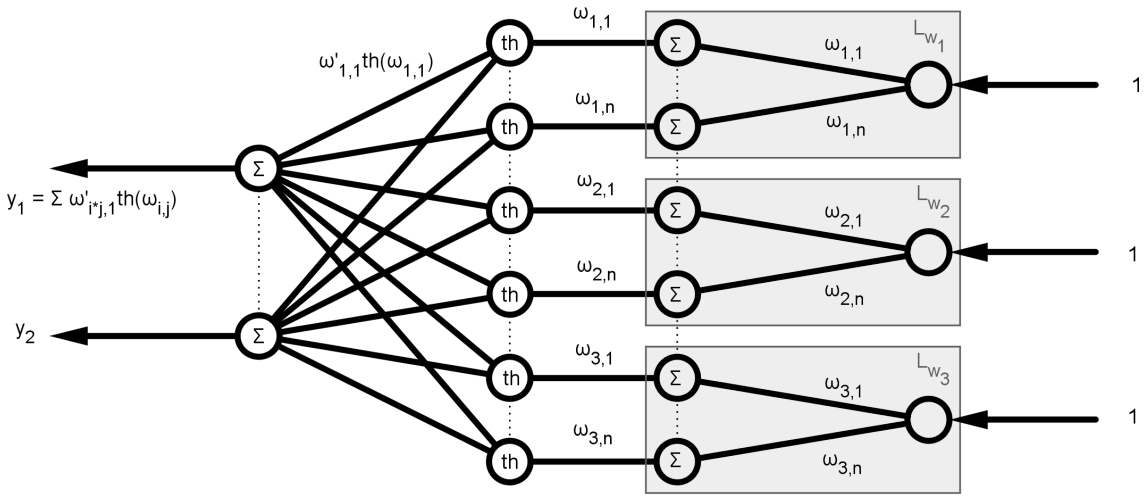

FIGURE 7 – Architecture *DMLP* équivalente au *MLP* décrit en FIGURE 5. Chaque mot $w_{i\in[1..3]}$ composant notre exemple d'apprentissage est désormais représenté par une couche linéaire L_{w_i}. Le vecteur de poids ω_i de cette couche étant le plongement associé au mot w_i considéré, ces derniers deviennent des paramètres du réseau. Lorsqu'on souhaite soumettre au réseau un nouvel exemple d'apprentissage $w'_{i\in[1..3]}$, il suffit de remplacer les vecteurs poids ω_i par les plongements ω'_i associés. Par soucis d'efficacité, cette opération est réalisée à l'aide d'un partage dynamique de paramètres entre notre couche d'entrée et notre bibliothèque L de couches linéaires contenant nos plongements raffinés au cœur de leur vecteur poids.

4.3 Utilisation de connaissances *a priori*

Nous désirons utiliser les connaissances *a priori* pouvant être extraites de notre corpus d'apprentissage afin de guider notre classification. Soit un mot w, nous nous concentrons sur les classes observées à chaque occurrence de w pour obtenir un *filtre* à appliquer aux prédictions. Ainsi, lors de la comparaison des scores des différentes classes, seules les classes observées sont mise en compétition. Dans le cas où w serait absent du corpus d'apprentissage, nous considérons l'ensemble des classes comme ayant été observées, ce qui revient à ne pas filtrer les prédictions de notre réseau de neurones.

5 Expérimentations & Résultats

Afin d'obtenir des performances directement comparables aux nôtres, nous avons réalisé une évaluation d'une implémentation des *Conditional random fields* (Nasr *et al.*, 2014) avec différents ensembles de *features*. Habituellement, à chaque mot sont associés :

– les *n-grams* le contenant pour n allant de 1 à 5 ;
– les *bigrams* d'étiquettes *POS* apportant une information de séquence ;
– ses attributs morphologiques (préfixes, suffixes ainsi que des *features* de présence de majuscules ou de symboles).

Model	All Words Accuracy	Unknown Words Accuracy
Conditional random field (3-gram uniquement)	95,55%	66,02%
Conditional random field (3-gram et toutes les *features*)	97,30%	87,59%
Conditional random field (5-gram et toutes les *features*)	97,43%	88,36%

TABLE 4 – Évaluations d'une implémentation des *CRF* pour le *POS tagging* (Nasr *et al.*, 2014) sous restrictions

Sur la première évaluation, les *features* sont restreintes aux *3-grams*. Ce système est par conséquent comparable au notre en terme d'information à disposition. La dernière, réalisée quant à elle sans limitations, égale nos résultats de références. On peut ainsi observer le gain notable apporté par l'ajout de *features* de séquence et de morphologie, notamment sur la gestion de mots inconnus.

d_{win}	n	Structure des couches cachées	*Learning Rate*
3 et 5	300	$((d_{win} * n) \times (d_{win} * 100))$	0.01

TABLE 5 – jeux d'hyper-paramètres considérés, où d_{win} et n représentent respectivement la taille des fenêtres de mots envoyées en entrée du réseau et la dimension des *embeddings* extraits par WORD2VEC. Nous avons choisi d'utiliser deux couches cachées afin de permettre au réseau d'extraire des représentations internes issues de la combinaison des $features$ propres à chaque mot de la fenêtre.

Dans l'ensemble, nos résultats expérimentaux sont significativement proches de nos références, cf. TABLE 2. L'approche *DMLP* obtient globalement de meilleurs résultats car elle permet un raffinage des plongements guidé par la tâche. Elle présente également de meilleures performances que le *CRF* dépourvu d'informations de séquence et morphologiques.

Toutefois, nos résultats se trouvent dans la plage basse des résultats de références car n'a été utilisé aucune information de séquence ou d'optimisation globale visant à prendre toute décision en se servant explicitement d'une décision prise antérieurement. En effet, dans nos approches les mots sont étiquetés un par un, indépendamment les uns des autres, sans utiliser, par exemple, les $tags$ prédits des mots précédents ou suivants. L'évaluation du *MLP* naïf quant à elle n'a pas été mené à son terme à cause du trop important temps de traitement qu'elle nécessiterait. Elle permet cependant une estimation de l'efficience de notre approche.

Model	All Words Accuracy	Unknown Words Accuracy	ms/exemple
3win-MLP	na	na	350ms
5win-MLP	na	na	973ms
3win-SMLP	95,87%	84,14%	1,09ms
5win-SMLP	96,09%	84,64%	3,11ms
3win-DMLP	96,51%	85,26%	1,19ms
5win-DMLP	96,79%	86,20%	3,33ms

TABLE 6 – Résultats expérimentaux

6 Conclusion

Issue de la théorie des réseaux de neurones profonds, l'approche que nous proposons égale nos résultats de référence sur le Penn TreeBank. Cette dernière se base sur l'extraction automatique de représentations vectorielles des mots à partir d'importants corpus non-annotés ainsi que sur les théories du *Deep Learning* permettant de raffiner, couche après couche, ces représentations internes afin qu'elles gagnent en pertinence tout au long de l'apprentissage.

Les performances prometteuses observées sur le *POS Tagging* confirment le bien-fondé de notre méthodologie n'utilisant pourtant aucune information de séquence, contrairement à nos résultats de référence. De plus, la comparaison des performances que nous avons obtenues à celles d'un *CRF* n'utilisant également aucune information de séquence met en évidence le potentiel de notre approche. En effet, il n'y a qu'un pas pour doter notre architecture d'une structure récurrente afin de lui permettre d'étiqueter simultanément tous les mots d'une même phrase. C'est ce que nous considèrerons dans de futurs travaux au même titre que l'ajout de *feature* morphologiques tels que les suffixes ou les préfixes.

C'est donc confiants que notre attention se porte désormais sur des tâches plus complexes du traitement automatique des langages naturels, telles que les traitements joints et la détection de disfluences dans des corpus oraux non-canoniques, afin de tester la robustesse de notre modèle au changement de domaine.

Remerciements

Ces travaux de recherche ont été financés en partie par l'Union Européenne à travers le projet SENSEI [1] (FP7/2007-2013 - n° 610916 – SENSEI).

1. http ://www.sensei-conversation.eu

Références

BRANTS T. (2000). TnT - A Statistical Part-of-Speech Tagger. In *Proceedings of the 6th Applied Natural Language Processing*, ANLC '00, p. 224–231 : Association for Computational Linguistics.

COLLOBERT R., KAVUKCUOGLU K. & FARABET C. (2012). Implementing Neural Networks Efficiently. In G. MONTAVON, G. ORR & K.-R. MULLER, Eds., *Neural Networks : Tricks of the Trade*.

COLLOBERT R., WESTON J., BOTTOU L., KARLEN M., KAVUKCUOGLU K. & KUKSA P. (2011). Natural Language Processing (Almost) from Scratch. In *the Journal of Machine Learning Research 12*, p. 2461–2505.

FISCHER A. & IGEL C. (2012). An introduction to restricted Boltzmann machines. In L. ALVAREZ, M. MEJAIL, L. GOMEZ, & J. JACOBO, Eds., *Proceedings of the 17th Iberoamerican Congress on Pattern Recognition, volume 7441 of LNCS*, p. 14–36.

GIESBRECHT E. & EVERT S. (2009). Part-of-Speech (POS) Tagging - a solved task ? An Evaluation of POS Taggers for the German Web as Corpus. In *Proceedings of the Fifth Web as Corpus Workshop (WAC5)*, p. 27–35, San Sebastian : Elhuyar Fundazioa.

GIMÉNEZ J. & MÁRQUEZ L. (2004). SVMTool : A general POS tagger generator based on Support Vector Machines. In *In Proceedings of the 4th International Conference on Language Resources and Evaluation*, p. 43–46.

MANNING C. (2011). Part-of-Speech Tagging from 97% to 100% : Is it time for some linguistics ? In A. GELBUKH, Ed., *Computational Linguistics and Intelligent Text Processing, 12th International Conference (CICLing), Part I. Lecture Notes in Computer Science 6608*, p. 171–189.

MIKOLOV T., CHEN K., CORRADO G. & DEAN J. (2013a). Efficient Estimation of Word Representations in Vector Space. volume abs/1301.3781.

MIKOLOV T., SUTSKEVER I., CHEN K., CORRADO G. & DEAN J. (2013b). Distributed Representations of Words and Phrases and their Compositionality. In *Deep Learning Workshop at the 2013 Conference on Neural Information Processing Systems (NIPS)*.

NASR A., BECHET F., FAVRE B., BAZILLON T., DEULOFEU J. & VALLI A. (2014). Automatically enriching spoken corpora with syntactic information for linguistic studies. In *LREC*.

TOUTANOVA K., MANNING C., KLEIN D. & SINGER Y. (2003). Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In *HLT-NAACL*.

TURIAN J., RATINOV L. & BENGIO Y. (2010). Word representations : a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, p. 384–394 : Association for Computational Linguistics.