Modèles de langue neuronaux: une comparaison de plusieurs stratégies d'apprentissage

Quoc-Khanh Do[1, 2] Alexandre Allauzen[1, 2] François Yvon[1]
(1) LIMSI/CNRS, rue John von Neumann, Campus Universitaire Orsay 91 403 Orsay
(2) Université Paris Sud, 91 403 Orsay
prenom.nom@limsi.fr

Résumé. Alors que l'importance des modèles neuronaux dans le domaine du traitement automatique des langues ne cesse de croître, les difficultés de leur apprentissage continue de freiner leur diffusion au sein de la communauté. Cet article étudie plusieurs stratégies, dont deux sont originales, pour estimer des modèles de langue neuronaux, en se focalisant sur l'ajustement du pas d'apprentissage. Les résultats expérimentaux montrent, d'une part, l'importance que revêt la conception de cette stratégie. D'autre part, le choix d'une stratégie appropriée permet d'apprendre efficacement des modèles de langue donnant lieu à des résultats à l'état de l'art en traduction automatique, avec un temps de calcul réduit et une faible influence des hyper-paramètres.

Abstract. If neural networks play an increasingly important role in natural language processing, training issues still hinder their dissemination in the community. This paper studies different learning strategies for neural language models (including two new strategies), focusing on the adaptation of the learning rate. Experimental results show the impact of the design of such strategy. Moreover, provided the choice of an appropriate training regime, it is possible to efficiently learn language models that achieves state of the art results in machine translation with a lower training time and a reduced impact of hyper-parameters.

Mots-clés : Réseaux de neurones, modèles de langue n-gramme, traduction automatique statistique.

Keywords: Neural networks, n-gram language models, statistical machine translation.

1 Introduction

Après quelques années de relatif désintérêt, les réseaux de neurones artificiels ont retrouvé une place centrale dans le paysage de l'apprentissage automatique et ont récemment permis des avancées significatives pour de nombreux domaines applicatifs. Pour ce qui concerne le traitement automatique des langues (TAL), les applications sont également nombreuses et variées. Historiquement, les modèles de langue neuronaux ont été une des premières réalisations marquantes, avec des applications en reconnaissance automatique de la parole (RAP), depuis les travaux pionniers de (Nakamura *et al.*, 1990) jusqu'aux développements ultérieurs de (Bengio *et al.*, 2003; Schwenk, 2007; Mnih & Hinton, 2007; Le *et al.*, 2011; Mikolov *et al.*, 2011). Les modèles neuronaux ont été également appliqués à d'autres tâches complexes de modélisation, comme par exemple l'analyse syntaxique (Socher *et al.*, 2013), l'estimation de similarité sémantique (Huang *et al.*, 2012), les modèles d'alignement de mots (Yang *et al.*, 2013) ou encore en traduction automatique statistique (TAS) (Le *et al.*, 2012a; Kalchbrenner & Blunsom, 2013).

Les modèles de langue continue de jouer un rôle prépondérant dans de nombreuses applications du TAL, depuis la reconnaissance vocale jusqu'à la correction d'orthographe ou encore la traduction automatique. Les approches usuelles reposent sur des modèles n-grammes discrets, estimés avec des méthodes de lissage (Chen & Goodman, 1998). Pour ces modèles, l'occurrence d'un mot dans son contexte est considérée comme la réalisation d'une variable aléatoire discrète, dont l'espace de réalisation est le vocabulaire tout entier et au sein duquel il n'existe aucune relation entre les mots. Par exemple, aucune information statistique n'est partagée entre formes morphologiquement ou sémantiquement apparentées. Le caractère très inégal des distributions d'occurrences dans les textes implique que les modèles résultants sont souvent estimés à partir de petits nombres d'occurrences et possèdent une faible capacité de généralisation. Par ailleurs, le nombre de paramètres d'un modèle n-gramme augmentant de manière exponentielle avec son ordre, il est illusoire d'espérer

résoudre ce problème en augmentant la quantité des ressources textuelles disponibles [1]. Par opposition, les modèles de langue neuronaux (Bengio *et al.*, 2003) se caractérisent par une méthode d'estimation alternative des qui se fonde sur une représentation *continue* [2], puisque chaque mot du vocabulaire est représenté comme un point dans un espace métrique. La probabilité n-gramme d'un mot est alors une fonction des représentations continues des mots qui composent son contexte. Ces représentations, ainsi que les paramètres de la fonction d'estimation, sont apprises conjointement par un réseau de neurones multi-couche ; une stratégie d'estimation qui permet que les mots partageant des similarités distributionnelles auront des représentations proches. Ainsi, ce type de modèle introduit la notion de similarité entre mots et son exploitation permet une meilleure exploitation des données textuelles. L'intégration de ce type de modèles a permis des améliorations systématiques et significatives des performances en RAP et en TAS (Schwenk, 2007; Le *et al.*, 2011, 2012a). De plus, les représentations continues peuvent servir à de nombreuses tâches, comme par exemple l'étiquetage en parties du discours et en rôle sémantique (voir (Turian *et al.*, 2010; Collobert *et al.*, 2011) pour une vue d'ensemble).

Malgré toutes ces qualités, le coût computationel des modèles neuronaux, tant à l'apprentissage qu'à l'inférence, ainsi que la difficulté à mettre en œuvre une stratégie d'apprentissage efficace limitent la diffusion de ce type d'approche. Plusieurs solutions ont été proposées afin de réduire ce coût, comme l'utilisation de *short-list* impliquant les modèles neuronaux uniquement pour les mots les plus fréquents (Schwenk, 2007), la modification du critère d'optimisation (Mnih & Teh, 2012), ou encore l'usage d'une couche de sortie structurée (Mnih & Hinton, 2008) comme le modèle SOUL (Le *et al.*, 2011). Ces approches partagent néanmoins le même algorithme d'estimation, qui repose sur la maximisation de l'entropie croisée, réalisée par descente de gradient stochastique. Si son implantation est relativement aisée, cet algorithme nécessite de fixer au préalable un certain nombre d'hyper-paramètres, qui ont une grande influence sur la vitesse de convergence et sur les performances finales. Or, le choix des hyper-paramètres reste très empirique et une part importante du coût computationel consommé lors de l'apprentissage est dû à l'exploration de différents jeux d'hyper-paramètres afin d'en garder le plus approprié. Cela peut en rebuter certains, tant il est vrai que fixer ces hyper-paramètres est complexe et relève d'une expertise difficile à décrire et à transmettre. Parmi ces hyper-paramètres, le choix du pas d'apprentissage (*learning rate*) est crucial puisqu'il permet d'effectuer un compromis entre la vitesse de convergence et les performances (Schaul *et al.*, 2012) : même si l'optimisation ne converge pas vers un meilleur modèle, une stratégie appropriée et adaptative pour régler ce pas permet d'accélérer grandement la convergence, réduisant d'autant le coût computationel.

L'objectif de cet article est, dans un premier temps, de décrire de manière unifiée les différentes stratégie d'ajustement du pas d'apprentissage de la littérature, et d'en proposer deux nouvelles. Dans un second temps, grâce aux résultats expérimentaux, nous souhaitons montrer que certaines stratégies peuvent être à la fois peu dépendantes des choix initiaux pour les hyper-paramètres et en même temps donner lieu à des performances état de l'art, tant en terme de perplexité qu'en terme de score BLEU lorsque le modèle est utilisé au sein d'un système de TAS. Le reste de l'article est organisé de la manière suivante : la section 2 décrit les modèles de langue neuronaux qui seront utilisés dans cette étude ; puis les différentes stratégies d'ajustement du pas d'apprentissage sont passées en revue à la section 3 ; les résultats expérimentaux sont enfin présentés à la section 4.

2 Modèles de langue neuronaux

On s'intéresse aux modèles de langue neuronaux de type n-grammes [3] dont les probabilités sont estimées dans un espace continu. Introduits par (Bengio *et al.*, 2003), ces modèles se distinguent des modèles discrets par leur aptitude à prendre en compte un large contexte. Il est ainsi possible d'entraîner un modèle 10-gramme, puisqu'ici le nombre de paramètres ne croît que linéairement avec l'ordre du modèle. En revanche, une limitation des modèles décrits par (Bengio *et al.*, 2003; Schwenk, 2007), est que la taille du vocabulaire de prédiction est limitée afin de rendre le temps de calcul acceptable. Le modèle SOUL (*Structured OUtput Layer*), proposé par (Le *et al.*, 2011) permet de lever cette contrainte grâce à une représentation hiérarchique du vocabulaire de sortie. C'est ce modèle que nous étudions dans cet article.

2.1 Le modèle neuronal standard

Un modèle n-gramme calcule itérativement la probabilité que le mot w_i apparaisse dans un contexte determiné par les $n-1$ mots $\mathrm{w}_{i-\mathrm{n}+1}^{i-1}$ qui le précèdent, soit $P(\mathrm{w}_i|\mathrm{w}_{i-\mathrm{n}+1}^{i-1})$. Le modèle neuronal standard, tel qu'il est décrit dans (Bengio

1. Disposer de plus gros corpus reste naturellement un moyen sûr d'améliorer les performances (Brants *et al.*, 2007).
2. Les modèles neuronaux sont souvent qualifiés de modèles continus.
3. Par opposition aux modèles *récurrents* (Mikolov *et al.*, 2011), dont le contexte est a priori illimité. Une comparaison récente entre ces deux approches a montré qu'elles obtiennent des performances voisines, mais que le modèle n-gramme passe plus facilement à l'échelle (Le *et al.*, 2012b).

et al., 2003), réalise ce calcul de la manière suivante. Chaque mot du contexte en entrée du réseau est encodé par un vecteur **w** de dimension $|V|$, où $|V|$ est la taille du vocabulaire ; **w** contient une seule valeur non-nulle (typiquement égale à 1), sa position dans le vecteur permettant de repérer le mot parmi le vocabulaire (voir la figure 1) [4]. Les mots du contexte sont alors projetés dans un espace continu de dimension m par multiplication avec la matrice **R** de dimension $|V| \times$ m. Cette étape consiste en fait à sélectionner, dans la matrice **R**, le vecteur continu représentant chaque mot du contexte. Ainsi, un mot du vocabulaire est représenté par un vecteur dense à valeurs dans $\mathbb{R}^m$. En pratique, m $\ll |V|$ (500 au lieu de quelques centaines de milliers), ce qui explique la réduction considérable du nombre de paramètres du modèle. Dans un deuxième temps, ces $n-1$ représentations continues sont concaténées pour former $\mathbf{i}^{(1)}$, la première couche cachée du réseau. Un réseau multi-couche peut être vu comme un empilement de couches neuronales, chaque couche se définissant par un vecteur d'entrée $\mathbf{i}^{(l)}$, sa matrice de connexion $\mathbf{W}^{(l)}$, un vecteur de biais associé $\mathbf{b}^{(l)}$ et un vecteur de sortie $\mathbf{o}^{(l)}$ calculé par :

$$\mathbf{o}^{(l)} = \sigma\left(\mathbf{W}^{(l)}\mathbf{i}^{(l)} + \mathbf{b}^{(l)}\right), \tag{1}$$

où l désigne le numéro de la couche et σ est une fonction d'activation non-linéaire qui s'applique à chaque composante d'un vecteur. La couche de sortie **o** possède un neurone par mot à prédire (w_i) et la probabilité qui lui est associée est calculée grâce à une fonction d'activation de type *softmax* :

$$\mathbb{P}_{\boldsymbol{\theta}}(\mathrm{w}_i|\mathrm{w}_{i-\mathrm{n}+1}^{i-1}) = \exp(\mathbf{o}_{\mathrm{w}_i}) / \sum_{\mathrm{w}\in\mathbf{V}} \exp(\mathbf{o}_{\mathrm{w}}). \tag{2}$$

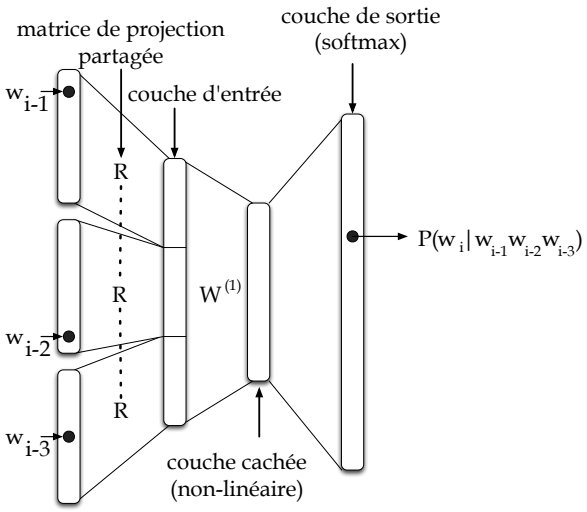


FIGURE 1: Architecture d'un modèle de langue neuronal standard pour $n = 4$

2.2 Le modèle SOUL

La comparaison des équations (1) et (2) aide à comprendre pourquoi la complexité d'un tel modèle se concentre dans la couche de sortie : le calcul d'une probabilité n-gramme nécessite d'abord une multiplication matricielle dont une des dimension est la taille du vocabulaire, puis la somme sur l'ensemble du vocabulaire pour normaliser la distribution n-gramme. Afin de réduire le temps de calcul, (Morin & Bengio, 2005; Mnih & Hinton, 2008) proposent de représenter le vocabulaire de sortie par un arbre binaire, représentant chaque mot par un chemin allant de la racine à une feuille. Dans cet arbre, chaque nœud interne représente une classe ou une sous-classe de mots. Le réseau de neurone peut alors servir à calculer itérativement la probabilité de ce chemin, plutôt que calculer directement celle du mot. Ces travaux ont été étendus par Le *et al.* (2011), qui proposent la structure SOUL (Figure 2). Cette structure se singularise par deux aspects : le modèle s'appuie sur un arbre d'arité quelconque et non simplement binaire ; une couche *softmax* est utilisée à chaque nœud de l'arbre au lieu d'une couche par niveau.

Compte-tenu des distributions très inégales des occurrences des mots, il reste souvent plus efficace de traiter à part les mots les plus fréquents. Ainsi, le modèle SOUL garde la notion de *short-list* regroupant les mots les plus fréquents du vocabulaire. Chacun de ces mots constitue une classe (singleton), comme représenté sur la partie droite de la figure 2.

4. Par la suite, les lettres en gras désignent un vecteur ou une matrice, selon qu'elles sont en minuscule ou majuscule.

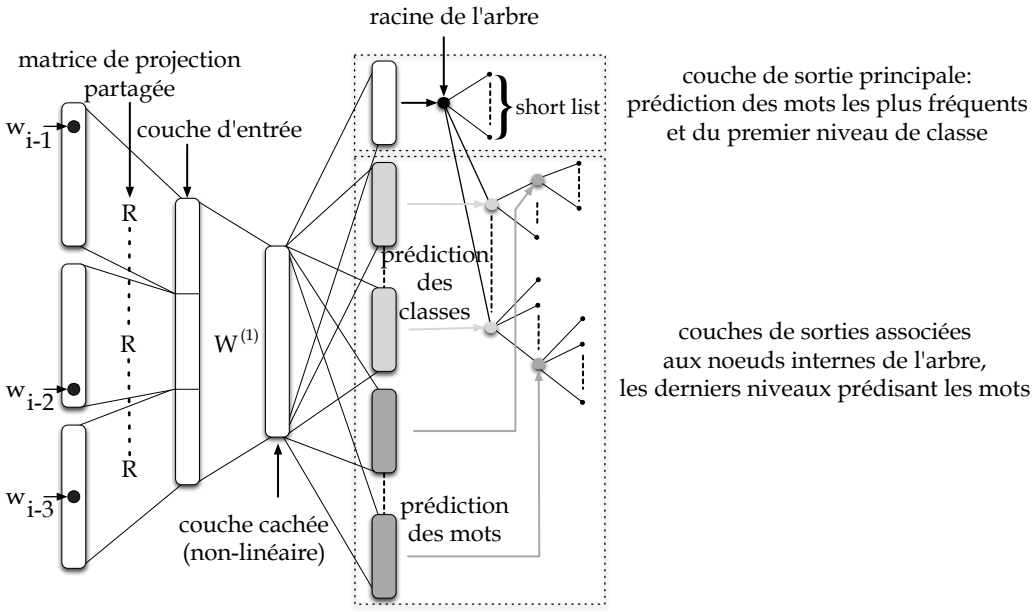


FIGURE 2: Architecture d'un modèle de langue SOUL pour $n = 4$

2.3 Apprentissage

L'apprentissage d'un modèle de langue neuronal se fonde sur la maximisation de la log-vraisemblance des paramètres ; l'objectif est alors de maximiser la fonction objectif définie ainsi :

$$\mathrm{L}(\boldsymbol{\theta}) = \sum_{\mathrm{w}_{i-\mathrm{n}+1}^{i} \in \mathbb{D}} \log \mathbb{P}_{\boldsymbol{\theta}}(\mathrm{w}_i | \mathrm{w}_{i-\mathrm{n}+1}^{i-1}) - \mathbb{R}(\boldsymbol{\theta}), \tag{3}$$

où $\boldsymbol{\theta}$ est le vecteur regroupant l'ensemble des paramètres du réseau neuronal : matrices de représentation et de connexion, ainsi que les biais. Le terme de régularisation (parfois nommé *weight decay*) $\mathbb{R}(\boldsymbol{\theta})$ correspond au carré de la norme euclidienne du vecteur de paramètres. Les termes de biais $\mathbf{b}^{(l)}$ de chaque couche l sont usuellement exclus de la régularisation. La méthode la plus utilisée pour maximiser (3) est la descente de gradient stochastique (Bottou, 2012). Le vecteur de paramètres $\boldsymbol{\theta}$ est mis à jour à l'instant $t+1$ de la manière suivante :

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \eta \nabla \mathrm{L}(\boldsymbol{\theta}^{(t)}), \tag{4}$$

où $\nabla \mathrm{L}(\boldsymbol{\theta}^{(t)})$ désigne le gradient de la fonction objectif pour la valeur des paramètres à l'instant t. Le pas d'apprentissage sur lequel nous focalisons notre étude est noté η. L'estimation du gradient et de la mise à jour peut se faire soit sur l'ensemble des données d'apprentissage (mode *batch*), soit sur quelques centaines d'exemples (mode *mini-batch*) ou bien pour chaque exemple (mode *online*). Par la suite nous considérons l'approche *mini-batch* qui offre un compromis entre garantie théorique et rapidité de convergence.

Les méthodes basées sur le Hessien demandent de calculer les dérivées secondes et ont été également envisagées pour l'apprentissage des réseaux neuronaux. Les algorithmes dit de Newton remplacent ainsi le pas d'apprentissage η dans l'équation (4) par l'inverse de la matrice Hessienne estimée à l'instant t :

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \left(\mathrm{H}^{(t)}\right)^{-1} \nabla \mathrm{L}(\boldsymbol{\theta}^{(t)}). \tag{5}$$

La portée de ce type d'approche pour les modèles neuronaux est limitée par un coût calculatoire exorbitant. En effet, ces méthodes requièrent l'estimation et le stockage de la matrice Hessienne qui est de dimension $|\boldsymbol{\theta}|^2$. À titre d'exemple, dans nos expériences, la matrice de projection $\mathbf{R}$ regroupe les représentations continues de dimension 500 pour chacun des $372K$ mots du vocabulaire et contient environ 1.8×10^8 paramètres. Par contre, les méthodes Quasi-Newton proposent une approximation du Hessien $\mathrm{H}^{(t)}$. Par exemple, une approximation diagonale de la matrice Hessienne équivaut à utiliser un pas d'apprentissage par paramètre. Ce type d'approche sera décrite dans la section suivante. Pour une vue d'ensemble de ces méthodes appliquées aux modèles neuronaux, le lecteur peut se référer à (LeCun *et al.*, 1998; Bengio, 2012).

L'apprentissage du modèle SOUL, s'il repose également sur la maximisation de la fonction de vraisemblance par des méthodes de gradient stochastique, implique en fait 3 étapes décrites dans (Le *et al.*, 2013) et que nous résumons ici :

1. apprentissage en quelques itérations [5] d'un modèle standard avec *short-list* : le vocabulaire de sortie est réduit aux quelques milliers de mots les plus fréquents. L'objectif est de disposer une première estimation des représentations continues des mots (**R**).
2. construction de l'arbre de partitionnement du vocabulaire. Pour cela, une version récursive de l'algorithme des K-moyennes est appliquée au vocabulaire, chaque mot étant représenté dans l'espace continu.
3. apprentissage du réseau global.

2.4 Représentations continues des mots et leur utilité

Afin de comprendre en quoi les représentations continues apprises (notées **R** à la section précédente) peuvent être utiles à différentes tâches du TAL, le tableau 1 décrit le voisinage de mots choisis au hasard, en français et en espagnol. Les représentations continues sont apprises par un modèle de langue SOUL utilisé lors des dernières campagnes d'évaluation WMT (voir pour plus de détails (Allauzen *et al.*, 2013)). Notons que les représentations sont apprises afin de modéliser le caractère distributionnel des mots puisqu'il s'agit d'un modèle n-gramme. Selon (Collobert *et al.*, 2011), la tâche d'apprentissage qui sert de supervision a un impact sur les représentations apprises et donc sur le type de similarités capturées. Une piste prometteuse de recherche pour apprendre ces représentations est de considérer différentes tâches, afin de pouvoir modéliser différentes caractéristiques linguistiques des mots.

mot	**voisins les plus proches**
évolution	décroissance - dynamisation - structuration - inflexion - atonie - fléchissement - infléchissement - hétérogénéité - réalignement
biotechnologie	nanotechnologie - photonique - bioinformatique - protéomique - nanotechnologies - microélectronique - géomatique - biotechnologies - microfinance
64	63 - 66 - 55 - 62 - 51 - 60 - 52 - 57 - 53
Renault	VW - Michelin - Nissan - Mercedes-Benz - Safran - Renault-Nissan - Faurecia - Volkswagen - Thalès
ouvre	ouvrait - ouvrent - ouvrira - ouvrir - ouvrirait - ouvraient - ouvrant - ouvriront - ouvrit
afiliados	agremiados - afiliadas - adheridos - afiliado - afiliada - adherentes - adscritas - empadronados - cotizantes
tristeza	angustia - desesperanza - desánimo - desilusión - amargura - asombro - estupor - resignación - incredulidad
debatiendo	discutiendo - examinando - debata - discutimos - debatimos - debatamos - debatiremos - debatió - debaten
Prado	Valme - Casal - Barranco - Atienza - Cabezas - Alcázar - Martirio - Eroski - Bermejales
Líbano	Argelia - Tíbet - Zimbabue - Jordania - Chechenia - Golán - Darfur - palestinas - OLP

TABLE 1: Exemples de voisinages dans l'espace de représentation continu pour des mots choisis aléatoirement, pour le français puis pour l'espagnol. Pour chaque mot considéré, le voisinage est défini ici par les 9 mots les plus proches.

3 Stratégies d'ajustement du pas d'apprentissage

Dans la plupart des cas, les modèles neuronaux utilisent un pas d'apprentissage dont la valeur initiale est fixée empiriquement, puis adaptée au cours de l'apprentissage via des stratégies simples. À notre connaissance, il n'existe pas de

5. Une itération correspond à un passage sur les données d'apprentissage. L'itération sert souvent d'unité de temps pour l'apprentissage des modèles neuronaux.

comparaisons de ces différentes stratégies qui permettrait d'évaluer leur impact sur les performances et sur le temps d'apprentissage. Pourtant, la littérature qui traite de la descente de gradient stochastique s'accorde sur le fait que le pas de gradient a un fort impact sur la vitesse de convergence (Robbins & Monro, 1951). De plus, une bonne stratégie d'ajustement permet de s'affranchir de l'influence de la valeur initiale sans altérer les performances du modèle (Schaul *et al.*, 2012). Ces stratégies d'ajustement peuvent se répartir en deux groupes : celles qui utilisent un pas d'apprentissage partagé par tous les paramètres, et d'autres qui ajustent un pas d'apprentissage par paramètre ou par groupe de paramètres.

3.1 Un pas d'apprentissage global

La stratégie la plus utilisée (**Power Scheduling**) impose la décroissance suivante : $\eta^{(t)} \sim t^{-1}$. Il a été montré que cette stratégie donnait la meilleure convergence asymptotique dans le cas d'un pas d'apprentissage global (Xu, 2011). De manière plus précise, l'instant t indique le nombre de mises à jour effectuées, et le pas d'apprentissage à l'instant t se calcule de la manière suivante : $\eta^{(t)} = \eta^{(0)}/(1 + \tau t)$, avec τ le taux de décroissance (*learning rate decay*). Dans (Bengio *et al.*, 2003), les auteurs fixent cet hyper-paramètre empiriquement à la valeur 10^{-8}. Le *et al.* (2012b) utilisent une variante de cette stratégie : $\eta^{(t)} = \eta^{(0)}/(1 + \tau \mathrm{N}_e^{(t)})$, avec $\mathrm{N}_e^{(t)}$ le nombre d'exemples d'apprentissage vu jusqu'à l'instant t.

La stratégie heuristique (**Down Scheduling**), qui est utilisée pour les modèles SOUL, est à la fois plus simple et plus efficace. Cette stratégie s'appuie sur la perplexité estimée sur un corpus de validation : lorsque la perplexité recommence à croître, le pas d'apprentissage est divisé par 2 après chaque itération. Une variante plus élaborée (**Adjust Scheduling**) et souvent nommée adaptative (e.g., (Ollivier, 2013)) est la suivante : après chaque itération, si la perplexité mesurée sur le corpus de validation augmente, les mises à jour de cette itération sont annulées et le pas d'apprentissage est divisé par 2 ; sinon le pas d'apprentissage est multiplié par 1,1. Ces deux stratégies permettent de réduire l'ensemble des hyper-paramètres au seul pas d'apprentissage initial. Enfin, la stratégie la plus simple (**Fix Scheduling**) consiste à choisir empiriquement une valeur du pas d'apprentissage qui restera fixe tout au long des itérations.

3.2 Ajustement du pas d'apprentissage pour chaque paramètre

Il est souvent recommandé d'utiliser différents pas d'apprentissage selon la partie du réseau impliquée (LeCun *et al.*, 1998; Bottou, 2012). Ce type d'approche peut être considérée comme un compromis entre la descente de gradient stochastique et les méthodes de Gauss-Newton ou Quasi-Newton. Cela revient en effet à approcher le Hessien par une matrice diagonale. Ainsi, LeCun *et al.* (1998) recommandent pour différentes tâches de classification l'usage des algorithmes de Gauss-Newton ou de Levenberg-Marquardt. Plus récemment, l'algorithme **AdaGrad** (Adaptive Gradient) (Duchi *et al.*, 2011) a montré une certaine efficacité dans plusieurs applications du TAL (Green *et al.*, 2013; Socher *et al.*, 2013). Dans sa forme originale, le Hessien est approché de la manière suivante :

$$\mathrm{H}^{(t)} = \left(\sum_{i=1}^{t} \nabla \mathrm{L}(\boldsymbol{\theta}^{(i-1)}) \nabla \mathrm{L}(\boldsymbol{\theta}^{(i-1)})^T \right)^{1/2}.$$

AdaGrad est souvent utilisé dans sa forme diagonale, car il est alors d'un point de vue calculatoire peu coûteux. Plus précisément, notons $\mathrm{g}_j^{(t-1)}$ le gradient de la fonction objectif par rapport au paramètre $\theta_j^{(t-1)}$ à l'instant $t-1$, le pas d'apprentissage à l'instant t associé au paramètre θ_j est calculé ainsi :

$$\eta_j^{(t)} = \frac{\eta^{(0)}}{\left(1 + \mathrm{G}_j^{(t)}\right)^{1/2}}, \tag{6}$$

$$\mathrm{G}_j^{(t)} = \sum_{i=1}^{t} \left(\mathrm{g}_j^{(i-1)}\right)^2, \tag{7}$$

où $\eta^{(0)}$ est la valeur initiale du pas d'apprentissage (commun à tous les paramètres) et $\mathrm{G}_j^{(t)}$ représente l'accumulation des gradients estimés par le passé ($\mathrm{g}_j^{(i-1)}, i = 1...t$).

3.3 Bloc-AdaGrad pour la structure SOUL

Afin d'adapter l'algorithme AdaGrad à la structure SOUL, nous en proposons une variante qui définit un pas d'apprentissage par groupe (ou bloc) de paramètres. Ce choix est motivé par l'objectif de réduction du temps de calcul. En effet, les opérations matricielles implémentées dans la bibliothèque BLAS jouent un rôle essentiel dans cette réduction. Nous choisissons les groupes de paramètres correspondant à ceux dont les gradients sont stockés dans une même matrice et sont calculés par une opération matricielle par BLAS ; tel que l'ajustement d'un pas d'apprentissage pour chacun de ces groupes n'ajoute qu'une complexité insignifiante comparé aux configurations où un seul pas d'apprentissage est utilisé pour l'ensemble de paramètres du réseau. L'ajustement d'un pas d'apprentissage pour chaque paramètre, comme proposé par AdaGrad, augmenterait considérablement le temps d'apprentissage. Dans cet algorithme, le terme η_j associé au paramètre θ_j dans l'équation (6) est remplacé par η_{b_j}, le pas d'apprentissage associé à tous les paramètres du bloc b_j :

$$\eta_{\mathrm{b}_j}^{(t)} = \frac{\eta^{(0)}}{\left(1 + \mathrm{G}_{\mathrm{b}_j}^{(t)}\right)^{1/2}} \tag{8}$$

$$\mathrm{G}_{\mathrm{b}_j}^{(t)} = \sum_{i=1}^{t} \left(\mathrm{g}_{\mathrm{b}_j}^{(i-1)}\right)^2 = \frac{1}{|\mathrm{b}_j|} \sum_{i=1}^{t} \sum_{k \in \mathrm{b}_j} \left(\mathrm{g}_k^{(i-1)}\right)^2 \tag{9}$$

Pour chaque couche cachée et pour chaque couche de sortie, deux blocs de paramètres sont définis : un pour la matrice de connexion et un pour le vecteur de biais. Pour la matrice de projection **R**, un bloc est défini pour chaque mot, soit pour chaque représentation continue. Cette séparation en blocs n'introduit pas de calcul supplémentaire, puisque les mises à jour sont également séparées lors de la descente stochastique de gradient.

4 Expériences

Afin d'analyser l'impact des différentes stratégies d'ajustement du pas d'apprentissage décrites à la section 3, une série d'expériences est menée sur la modélisation de l'espagnol avec, comme application, la tâche de traduction automatique de l'anglais vers l'espagnol, telle qu'elle est définie par la campagne d'évaluation WMT 2013 [6]. Différents critères sont utilisés pour l'évaluation : la perplexité du modèle de langue mesurée sur un corpus de validation, la vitesse de convergence en nombre d'itération, et la dépendance de ces critères aux choix des hyper-paramètres. Le modèle neuronal peut également être évalué de manière extrinsèque lorsqu'il est utilisé comme modèle additionnel en traduction automatique.

Nous utilisons ici un modèle neuronal 10-gramme utilisant le même vocabulaire (372K mots) que le modèle conventionnel discret. Le système de traduction utilisé est décrit dans (Allauzen *et al.*, 2013), les résultats obtenus le placent parmi les meilleurs systèmes pour cette tâche.

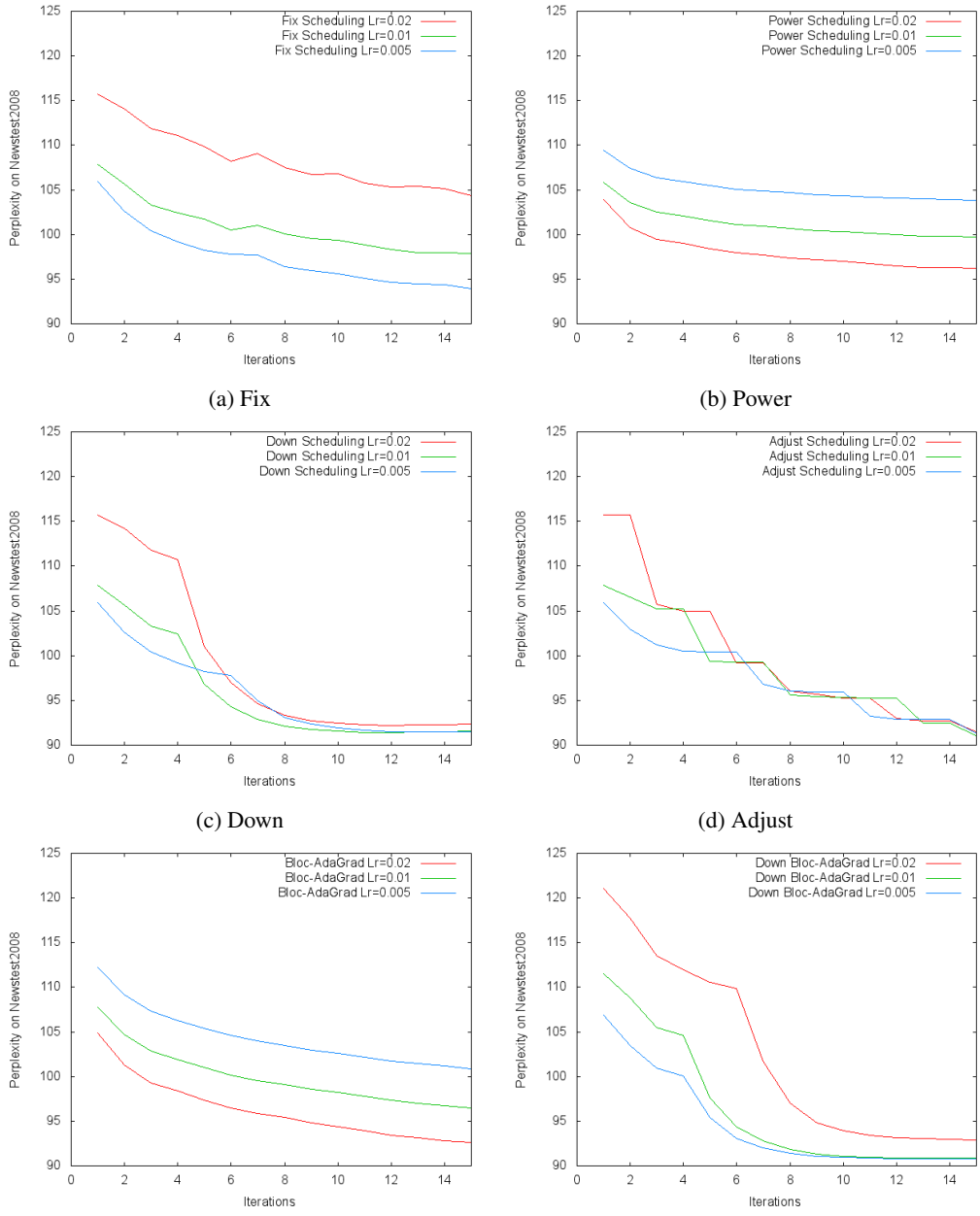
Tous les textes monolingues disponibles ont été utilisés pour l'apprentissage, soit 1.5 milliard de mots au total. L'apprentissage du réseau de neurone est itératif : à chaque itération, une sous-partie des données est échantillonnée aléatoirement, soit 15 millions de n-grammes dans nos expériences, puis divisée en paquet (mini-batch) de 128 n-grammes. Chaque paquet donne lieu à une mise à jour des paramètres du réseau. La perplexité est calculée à la fin de chaque itération sur le jeu de test *Newstest2008*. Afin de garantir que les différentes stratégies évaluées utilisent les mêmes données d'apprentissage, l'échantillon des données utilisé à chaque itération est conservé et partagé.

La procédure d'apprentissage du modèle SOUL est décrite dans (Le *et al.*, 2013) et résumée à la section 2.2. Les expériences suivantes concernent uniquement la dernière phase d'apprentissage des modèles SOUL, dont la configuration est la suivante : l'espace de projection est de dimension 500 ; le réseau contient deux couches cachées de dimensions respectives 1000 et 500 et la *short-list* contient les 2000 mots les plus fréquents.

4.1 Dépendance au choix des hyper-paramètres

Pour chaque stratégie d'ajustement, différents apprentissages ont été effectués en faisant varier la valeur initiale du pas d'apprentissage $\eta^{(0)}$. Les résultats en terme de perplexité sont représentés sur les figures 3a-3e. Sur ces courbes nous

6. http://www.statmt.org/wmt13/translation-task.html

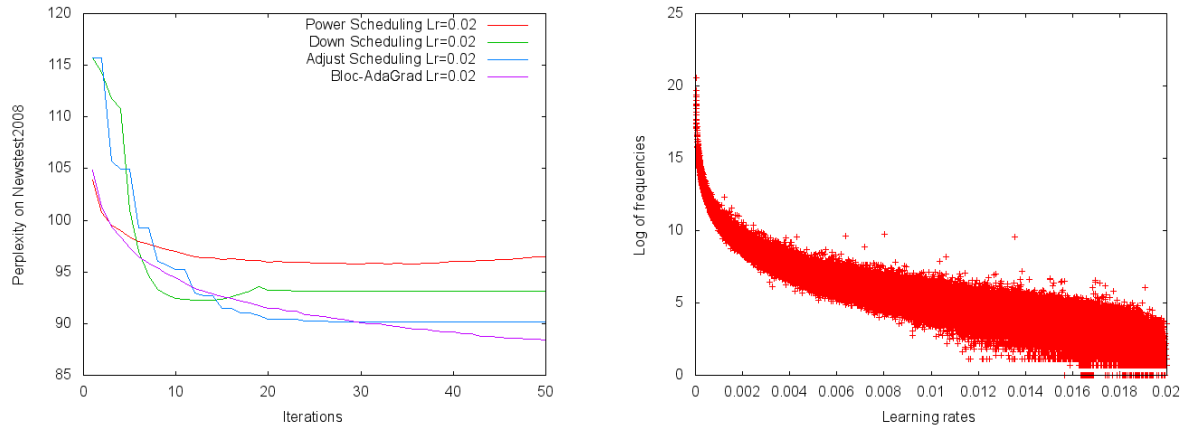


(a) Fix (b) Power

(c) Down (d) Adjust

(e) Bloc-AdaGrad (f) Down-Bloc-AdaGrad

FIGURE 3: Perplexités mesurées sur *Newstest2008* pour chacune des 6 stratégies décrites dans cet article. Dans le cas des figures 3c et 3f, chaque courbe d'apprentissage montre un point d'inflexion qui correspond à l'itération à laquelle le pas d'apprentissage est divisé par deux.



(a) Perplexités mesurées sur *Newstest2008* sur 50 itérations pour 4 stratégies d'ajustement. Pour la stratégie *Down*, on observe un phénomène de sur-apprentissage aux alentours de la 12ème itération.

(b) Relation entre la fréquence (échelle logarithmique) des mots et le pas d'apprentissage associé à leur représentation continue pour la stratégie *Bloc-AdaGrad*, après 40 itérations d'apprentissage.

FIGURE 4

pouvons observer que les performances obtenues avec les stratégies nommées *Fix*, *Power* and *Bloc-AdaGrad* sont très sensibles au choix de $\eta^{(0)}$, alors que les stratégies *Down* et *Adjust* mènent à des résultats plutôt stables au bout des 15 iterations, et ce avec différentes valeurs de $\eta^{(0)}$. Remarquons également que les deux dernières stratégies convergent nettement plus rapidement vers une perplexité basse.

De plus, dans le cas de la stratégie *Power*, la difficulté est augmentée par la présence d'un hyper-paramètre supplémentaire (le *learning rate decay*) τ. Dans toutes ces expériences, τ est fixé empiriquement à la valeur 5.10^{-7}. Néanmoins, cet hyper-paramètre a aussi une forte influence sur la vitesse de convergence.

Down-Bloc-AdaGrad : La forte dépendance au choix de η^0 de la méthode *AdaGrad* est un des défaut de cette méthode, puisque la recherche du paramètre optimal rend cette méthode coûteuse en terme de temps de calcul. Des travaux récents ont tenté de réduire cette dépendance, voir par exemple (Senior *et al.*, 2013). Appliqué au modèle SOUL, nous proposons une approche plus simple combinant la stratégie *Bloc-AdaGrad* avec la stratégie *Down*, qui est plus stable : d'une part, le gradient accumulé pour la première couche de sortie sert de valeur de normalisation, et d'autre part son influence est pondérée par le coefficient $\gamma^{(t)}$ dont la valeur est ajustée par la stratégie *Down*. Cette méthode est nommée *Down-Bloc-AdaGrad* : à l'instant t, la valeur du pas d'apprentissage pour le bloc b_j est calculée de la manière suivante :

$$\eta_{\mathrm{b}_j}^{(t)} = \frac{\gamma^{(t)} \times \left(1 + \mathrm{G}_{\mathrm{b}^*}^{(t)}\right)^{1/2}}{\left(1 + \mathrm{G}_{\mathrm{b}_j}^{(t)}\right)^{1/2}}$$

où b^* désigne le bloc de paramètres de la couche de sortie principale. Comme le montre la figure 3f, cette stratégie permet d'accélérer la vitesse de convergence tout en réduisant la dépendance au pas d'apprentissage initial, noté ici $\gamma^{(0)}$.

4.2 Apprentissage sur le long terme

Afin de comparer le comportement de différentes stratégies dans le cadre d'un apprentissage plus long, la figure 4a représente l'évolution de la perplexité au cours de 50 itérations d'apprentissage. Ces courbes suggèrent que seule la stratégie *Bloc-AdaGrad* peut tirer profit d'un apprentissage long, alors que les autres arrivent à stabilité à plus moins court terme. En particulier, les stratégies *Down* et *Power* donnent naturellement des courbes de perplexité qui se stabilisent rapidement, puisque le pas d'apprentissage tend rapidement vers 0. Au contraire, la stratégie *Bloc-AdaGrad* continue sa progression jusqu'au bout, puisque seuls les paramètres appartenant à un bloc fréquemment actif voient leurs pas d'apprentissage tendre vers 0. Ce phénomène concerne plus particulièrement les représentations continues des mots comme l'illustre la figure 4b. En effet les paramètres définissant la représentations continu d'un mot forment un bloc. Ainsi pour un mot fréquent, sa représentation continue sera fréquemment mise à jour et son pas d'apprentissage tendra rapidement vers 0,

comme l'exprime l'équation (8). Au contraire, les représentations continues des mots rares peuvent bénéficier d'un temps d'apprentissage plus long.

De plus, une autre considération complète l'analyse du comportement de la stratégie *Bloc-AdaGrad*. En autorisant différents pas d'apprentissage, cette stratégie donne une plus grande liberté à l'algorithme de descente de gradient pour choisir la direction à chaque étape. En effet, le pas d'apprentissage idéal à un instant donné doit tenir compte de la courbure de la fonction à optimiser : dans une zone à forte courbure, le gradient peut évoluer rapidement pour certains paramètres, il est donc alors préférable pour ces paramètres de réduire le pas d'apprentissage et d'effectuer des mises à jour fréquentes afin de pouvoir suivre ces variations rapides. Dans le cas d'une stratégie globale d'ajustement du pas d'apprentissage, les degrés de liberté manquent et, même si le pas d'apprentissage tend vers 0, il est impossible de nuancer le pas d'apprentissage par composante afin de suivre efficacement la fonction à optimiser. En dépit de ces propriétés, la différence en perplexité entre les stratégies globale *Bloc-AdaGrad* au bout de 50 itérations reste somme toute faible. Une interprétation est le peu de mots rares présent dans le corpus *Newstest2008*, alors que justement la particularité de la stratégie *Bloc-AdaGrad* réside dans la manière de traiter les mots rares. En pratique, il est rarement possible, ni souhaitable, d'utiliser un régime d'apprentissage long, bien sûr à cause du temps de calcul mais aussi car cela se révèle souvent inutile. Ainsi on observe qu'avec la stratégie *Bloc-AdaGrad*, en multipliant le temps d'apprentissage par 3, la perplexité baisse seulement de 3 points (comparaison entre l'itération 15 et 45 sur la figure 4a). Une telle différence de perplexité n'est susceptible de modifier qu'à la marge les performances d'un système de TAS ou de RAP.

La figure 4a justifie également pour la stratégie *Down* le recours à un apprentissage plus court (*early-stopping*) afin d'éviter le sur-apprentissage. On observe en effet que le modèle atteint sa plus basse perplexité aux alentours de la 10ème itération ; ensuite, on observe une évolution typique de sur-apprentissage : la perplexité mesurée lors de l'apprentissage (non représentée ici) continue de diminuer alors que celle mesurée sur un autre jeu de données commence à croître. Notons cependant qu'un apprentissage long, bien que coûteux, peut être souhaitable, afin de justement détecter, selon les stratégies employées, le phénomène de sur-apprentissage (Bengio, 2012). De plus, dans le cadre de certaines applications, les données d'apprentissage peuvent arriver au fil du temps, de manière « illimitée ». Dans ce cas, il est crucial de choisir une stratégie capable d'apprendre sur le long terme. À ce titre, la stratégie *Bloc-AdaGrad* semble la plus appropriée.

4.3 Impact sur les résultats en traduction automatique

Afin de mesurer l'impact de ces stratégies sur les performances d'un système de traduction automatique, certains des modèles sont utilisés au sein d'un système état de l'art pour la tâche de WMT13 de traduction de l'anglais vers l'espagnol. Étant donné le coût computationel des modèles de langue neuronaux, ces modèles sont utilisés en post traitement. Le système de traduction est utilisé dans un premier temps afin de générer les k meilleures hypothèses de traduction, $k = 300$ dans les expériences présentées dans cet article. Puis, dans un second temps le modèle de langue neuronal est utilisé afin d'évaluer chacune des hypothèses : le score produit par le modèle de langue neuronal est ajouté aux autres scores utilisés par le système de traduction ; puis les hypothèses sont retriées en utilisant des poids pour chacun des modèles qui ont été optimisés sur des données de développement (*Newstest2011* dans notre cas). Cette optimisation utilise l'algorithme KBMIRA (*k-best Batch Margin Infused Relaxed Algorithm*) décrit par Cherry & Foster (2012).

Les résultats sont regroupés dans le tableau 2. Les performances de traduction sont évaluées en terme de score BLEU mesuré sur l'ensemble de développement (*Newstest2011*) et deux jeux de données de test : *Newstest2012* et *Newstest2013*. Ces résultats montrent qu'il y a qu'une faible différence entre un modèle entraîné avec la stratégie *Bloc-AdaGrad* pendant 50 itérations et un modèle entraîné avec les deux stratégies *Down* et *Down-Bloc-AdaGrad* après 8 itérations. Ce résultat confirme l'intérêt d'utiliser un apprentissage abrégé si la stratégie d'ajustement est correctement choisie, et les stratégies du type *Down*, dont *Down-Bloc-AdaGrad* fait partie, semble les meilleures adaptées. De plus remarquons la dépendance marquée de la stratégie *Fix* au choix du pas d'apprentissage initial. Même si les différences ici sont plutôt mineurs, nous insistons sur le fait que notre intérêt dans le choix des stratégies se trouve dans le temps d'entraînement (le nombre d'itérations) et la stabilité du système par rapport au choix de paramètres (ici, le pas d'apprentissage initial).

5 Conclusions

Dans cet article, nous avons présenté une étude comparative de différentes stratégies d'ajustement du pas d'apprentissage utilisé par les modèles de langue neuronaux. Cet hyper-paramètre a, comme le montrent les résultats expérimentaux, une grande influence sur la vitesse de convergence et sur les performances du modèle. Comme cadre expérimental, nous

	Perplexité *Nt08*	dev *Nt11*	*Nt12*	*Nt13*
Sans modèle neuronal	-	32,32	33,86	29,79
Fix $\eta = 0,02$ ite. 15	104,4	32,80	34,12	29,97
Fix $\eta = 0,01$ ite. 15	97,8	32,93	34,30	30,18
Fix $\eta = 0,005$ ite. 15	93,9	32,98	34,37	30,20
Down $\eta^0 = 0,005$ ite. 8	93,1	33,01	34,35	30,11
Down-Bloc-AdaGrad $\gamma^{(0)} = 0,005$ ite. 8	91,4	33,05	34,43	30,06
Bloc-AdaGrad ite. 50 $\eta^0 = 0,02$	88,4	-	34,48	30,12

TABLE 2: BLEU scores mesurés sur *Newstest2012* et *Newstest2013*.

nous sommes intéressés à la tâche de traduction de WMT13, de l'anglais vers l'espagnol. Les modèles de langue neuronaux présentés utilisent l'architecture SOUL qui a pour caractéristique de pouvoir prendre en compte un vocabulaire de prédiction aussi grand que nécessaire. Outre les stratégies existantes, nous avons proposé deux nouvelles stratégies, nommées respectivement *Bloc-AdaGrad* et *Down-Bloc-AdaGrad*. Ces stratégies permettent de s'adapter au modèle étudié par leur capacité à ajuster le pas d'apprentissage, non pas de manière uniforme pour l'ensemble des paramètres, mais par groupe (ou bloc) de paramètres. Chacun de ces blocs regroupe les paramètres d'une même couche, ou ceux associés à la représentation continue d'un mot du vocabulaire.

Les résultats expérimentaux montrent que certaines des stratégies étudiées sont très sensibles au choix du pas d'apprentissage initial. De plus la méthode *Down-Bloc-AdaGrad* se distingue des autres en permettant d'obtenir une convergence rapide vers une perplexité basse, et ce, avec une insensibilité relative au choix du pas d'apprentissage initial. Dans de futurs travaux, nous pensons qu'utiliser cette stratégie dès la première étape d'apprentissage du modèle SOUL donnera des améliorations plus marquées.

Références

ALLAUZEN A., PÉCHEUX N., DO Q. K., DINARELLI M., LAVERGNE T., MAX A., LE H.-S. & YVON F. (2013). LIMSI @ WMT13. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, p. 62–69, Sofia, Bulgaria : Association for Computational Linguistics.

BENGIO Y. (2012). Practical recommendations for gradient-based training of deep architectures. In G. MONTAVON, G. B. ORR & K.-R. MÜLLER, Eds., *Neural Networks : Tricks of the Trade (2nd ed.)*, volume 7700 of *Lecture Notes in Computer Science*, p. 437–478. Springer.

BENGIO Y., DUCHARME R., VINCENT P. & JAUVIN C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, **3**(6), 1137–1155.

BOTTOU L. (2012). Stochastic Gradient Descent Tricks. In G. MONTAVON, G. ORR & K.-R. MÜLLER, Eds., *Neural Networks : Tricks of the Trade*, volume 7700 of *Lecture Notes in Computer Science*, p. 421–436. Springer Berlin Heidelberg.

BRANTS T., POPAT A. C., XU P., OCH F. J. & DEAN J. (2007). Large language models in machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, p. 858–867.

CHEN S. F. & GOODMAN J. T. (1998). *An Empirical Study of Smoothing Techniques for Language Modeling*. Rapport interne TR-10-98, Computer Science Group, Harvard University.

CHERRY C. & FOSTER G. (2012). Batch tuning strategies for statistical machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, p. 427–436 : Association for Computational Linguistics.

COLLOBERT R., WESTON J., BOTTOU L., KARLEN M., KAVUKCUOGLU K. & KUKSA P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, **12**, 2493–2537.

DUCHI J., HAZAN E. & SINGER Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, **12**, 2121–2159.

GREEN S., WANG S., CER D. & MANNING C. D. (2013). Fast and adaptive online training of feature-rich translation models. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL)*, p. 311–321, Sofia, Bulgaria : Association for Computational Linguistics.

HUANG E., SOCHER R., MANNING C. & NG A. (2012). Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, p. 873–882, Jeju Island, Korea : Association for Computational Linguistics.

KALCHBRENNER N. & BLUNSOM P. (2013). Recurrent continuous translation models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, p. 1700–1709, Seattle, Washington, USA : Association for Computational Linguistics.

LE H.-S., ALLAUZEN A. & YVON F. (2012a). Continuous space translation models with neural networks. In *Proceedings of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies (NAACL-HLT)*, p. 39–48, Montréal, Canada : Association for Computational Linguistics.

LE H.-S., ALLAUZEN A. & YVON F. (2012b). Measuring the influence of long range dependencies with neural network language models. In *Proceedings of the NAACL-HLT 2012 Workshop : Will We Ever Really Replace the N-gram Model ? On the Future of Language Modeling for HLT*, p. 1–10, Montréal, Canada.

LE H.-S., OPARIN I., ALLAUZEN A., GAUVAIN J.-L. & YVON F. (2011). Structured output layer neural network language model. In *Proc. of ICASSP*, p. 5524–5527.

LE H. S., OPARIN I., ALLAUZEN A., GAUVAIN J.-L. & YVON F. (2013). Structured output layer neural network language models for speech recognition. *IEEE Transactions on Acoustic, Speech and Signal Processing*, **21**(1), 197 – 206.

LECUN Y., BOTTOU L., ORR G. & MULLER K. (1998). Efficient backprop. In G. ORR & M. K., Eds., *Neural Networks : Tricks of the trade* : Springer.

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

MNIH A. & HINTON G. E. (2007). Three new graphical models for statistical language modelling. In *ICML*, p. 641–648.

MNIH A. & HINTON G. E. (2008). A scalable hierarchical distributed language model. In D. KOLLER, D. SCHUURMANS, Y. BENGIO & L. BOTTOU, Eds., *Advances in Neural Information Processing Systems 21*, volume 21, p. 1081–1088.

MNIH A. & TEH Y. W. (2012). A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the International Conference on Machine Learning*.

MORIN F. & BENGIO Y. (2005). Hierarchical probabilistic neural network language model. In *AISTATS'05*, p. 246–252.

NAKAMURA M., MARUYAMA K., KAWABATA T. & KIYOHIRO S. (1990). Neural network approach to word category prediction for english texts. In *Proceedings of the 13th conference on Computational linguistics (COLING)*, volume 3, p. 213–218.

OLLIVIER Y. (2013). Riemannian metrics for neural networks. *CoRR*, **abs/1303.0818**.

ROBBINS H. & MONRO S. (1951). A stochastic approximation method. *Annals of Mathematical Statistics*, **22**, 400–407.

SCHAUL T., ZHANG S. & LECUN Y. (2012). No more pesky learning rates. preprint arXiv :1206.1106.

SCHWENK H. (2007). Continuous space language models. *Computer Speech and Language*, **21**(3), 492–518.

SENIOR A. W., HEIGOLD G., RANZATO M. & YANG K. (2013). An empirical study of learning rates in deep neural networks for speech recognition. In *Proc. of ICASSP*, p. 6724–6728.

SOCHER R., BAUER J., MANNING C. D. & ANDREW Y. N. (2013). Parsing with compositional vector grammars. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL)*, p. 455–465, Sofia, Bulgaria.

TURIAN J., RATINOV L.-A. & 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, Uppsala, Sweden : Association for Computational Linguistics.

XU W. (2011). Towards optimal one pass large scale learning with averaged stochastic gradient descent. *CoRR*, **abs/1107.2490**.

YANG N., LIU S., LI M., ZHOU M. & YU N. (2013). Word alignment modeling with context dependent deep neural network. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL)*, p. 166–175, Sofia, Bulgaria.