HAL open science

Représentations continues dérivées des caractères pour un modèle de langue neuronal à vocabulaire ouvert

Matthieu Labeau, Alexandre Allauzen

To cite this version:

Matthieu Labeau, Alexandre Allauzen. Représentations continues dérivées des caractères pour un modèle de langue neuronal à vocabulaire ouvert. TALN-RECITAL 2017, Jun 2017, Orléans, France. hal-02912472

HAL Id: hal-02912472

https://hal.science/hal-02912472v1

Submitted on 5 Aug 2020

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.

Représentations continues dérivées des caractères pour un modèle de langue neuronal à vocabulaire ouvert

Matthieu Labeau Alexandre Allauzen
Université Paris-Sud, LIMSI-CNRS, Univ. Paris-Saclay Orsay, France
labeau@limsi.fr, allauzen@limsi.fr

RÉSUMÉ

Cet article propose une architecture neuronale pour un modèle de langue à vocabulaire ouvert. Les représentations continues des mots sont calculées à la volée à partir des caractères les composant, grâce à une couche convolutionnelle suivie d'une couche de regroupement (*pooling*). Cela permet au modèle de représenter n'importe quel mot, qu'il fasse partie du contexte ou soit évalué pour la prédiction. La fonction objectif est dérivée de l'estimation contrastive bruitée (*Noise Contrastive Estimation*, ou NCE), calculable dans notre cas sans vocabulaire. Nous évaluons la capacité de notre modèle à construire des représentations continues de mots inconnus sur la tâche de traduction automatique IWSLT-2016, de l'Anglais vers le Tchèque, en ré-évaluant les N meilleures hypothèses (*N-best reranking*). Les résultats expérimentaux permettent des gains jusqu'à 0,7 point BLEU. Ils montrent aussi la difficulté d'utiliser des représentations dérivées des caractères pour la prédiction.

ABSTRACT

Opening the vocabulary of neural language models with character-level word representations

This paper introduces an architecture for an open-vocabulary neural language model. Word representations are computed on-the-fly by a convolution network followed by pooling layer. This allows the model to consider any word, in the context or for the prediction. The training objective is derived from the Noise-Contrastive Estimation to adapt it the open vocabulary case. We test the ability of our model to build representations of unknown words on the MT task of IWSLT-2016 from English to Czech, in a reranking setting. Experimental results show a gain up to 0.7 BLEU point. They also emphasize the difficulty and instability when training such models with character-based representations for the predicted words.

MOTS-CLÉS : Modèle de langue neuronal, Représentations continues dérivées des caractères, Traduction automatique par approche statistique.

KEYWORDS: Neural language model, Character-based representation, Statistical Machine Translation.

1 Introduction

La plupart des modèles de langue neuronaux, tels que les modèles n-grammes, (Bengio *et al.*, 2003) modélisent les mots par des vecteurs de valeurs réelles. Ils reposent donc sur la définition préalable d'un vocabulaire fini $\mathcal{V}$, qui contiendra la liste des mots dont le modèle paramétrise la représentation. Ainsi, une table de paramètres (*Look-up table*) est associée à $\mathcal{V}$, qui contiendra un ensemble de vecteurs réels de dimension d_r correspondant à chaque mot $w \in \mathcal{V}$, regroupés dans la

matrice $\mathbf{L} \in \mathbb{R}^{|\mathcal{V}|*d_r}$.

Bien que cette approche se soit révélée fructueuse quand appliquée à différentes tâches et langues, comme par exemple en reconnaissance de la parole (Schwenk, 2007), et en traduction automatique (Le *et al.*, 2012; Devlin *et al.*, 2014; Bahdanau *et al.*, 2014), elle reste limitée dans certains cas. Ainsi pour les langues morphologiquement riches, comme le Tchèque ou l'Allemand, couvrir l'ensemble du lexique se révèle difficile, à cause de l'explosion combinatoire du nombre de mots, la plupart n'apparaissant que rarement, voire même jamais, dans les données d'entraînement. Augmenter la taille du vocabulaire en conséquence ne ferait qu'augmenter le nombre de paramètres du modèle, alors qu'à données d'apprentissage constantes, le caractère parcimonieux des distributions de mots dans les textes ne fera qu'accroître les difficultés d'apprentissage. Une solution consiste à remplacer les mots rares par des classes dédiées, permettant ainsi de regrouper différents mots et ainsi améliorer l'estimation des probabilités. Néanmoins, utiliser des classes différentes pour gérer ces mots (Allauzen & Gauvain, 2005) ne résout que partiellement le problème, puisque ces mots sont par essence difficiles à classer.

De plus, pour la plupart des formes fléchies ou agglutinantes, ainsi que pour les mots composés, la structure est ignorée : le modèle attribue des paramètres à la modélisation de ce qui pourrait plus efficacement être appréhendé en décomposant le mot. Si elle peut améliorer le pouvoir de généralisation des modèles de langues neuronaux, l'utilisation de sous-mots comme unités (Botha & Blunsom, 2014; Sennrich *et al.*, 2016) dépend de méthodes permettant de les induire efficacement.

Nous proposons d'apprendre un modèle de langue dont l'unité est le mot, mais au vocabulaire ouvert. La difficulté principale est alors de définir une fonction objectif qui ne repose pas sur la définition préalable d'un vocabulaire fini, comme par exemple le maximum de vraisemblance. Comme décrit à la section 2, nous cherchons à construire des représentations continues des mots à la volée, à partir des séquences de leurs caractères, à l'aide de filtres de convolution qui, de manière implicite, capturent les régularités morphologiques. L'architecture du modèle est inspirée du modèle de langue n-gramme neuronal (Bengio *et al.*, 2003). Cependant, l'utilisation d'un vocabulaire ouvert est applicable à n'importe quel autre type de modèle de langue. Pour éviter la contrainte de la normalisation des probabilités de prédiction sur l'ensemble d'un vocabulaire, nous utilisons une fonction objectif similaire à l'estimation constrastive bruitée (Gutmann & Hyvärinen, 2012), ce qui permet au modèle de considérer un vocabulaire potentiellement infini. Afin d'évaluer l'efficacité de cette approche, nous utilisons notre modèle dans le contexte d'une tâche de traduction automatique à grande échelle, pour ré-évaluer les meilleures hypothèses proposées par le système de traduction, dans la section 3. Les résultats expérimentaux sont présentés dans la section 4.

2 Description du modèle

Habituellement, les représentations continues des mots apprises par le modèle sont des vecteurs de paramètres (*embeddings*), stockés dans une matrice $\mathbf{L}$. La représentation $\mathbf{r}_w^{word}$ d'un mot w est simplement la colonne de $\mathbf{L}$ correspondant à son indice dans le vocabulaire :

$$\mathbf{r}_w^{word} = [\mathbf{L}]_w$$

Cette première méthode est illustrée figure 1, dans l'encadré rouge.

2.1 Représentations continues de mots dérivées des caractères

Pour obtenir la représentation continue d'un mot à partir de celles de ses caractères, on utilise une *couche de convolution* (Waibel *et al.*, 1990; Collobert *et al.*, 2011). Un mot w est une séquence de caractères $\{c_1, .., c_{|w|}\}$, dont les représentations continues sont $\{[\mathbf{C}]_{c_1}, .., [\mathbf{C}]_{c_{|w|}}\}$, ou l'on note $[\mathbf{C}]_{c_i}$ le vecteur associé au caractère c_i. On applique un filtre de convolution $\mathbf{W}^{conv} \in \mathbb{R}^{d_r} \times \mathbb{R}^{d_c * n_c}$ à une fenêtre glissante de n_c caractères, ce qui produit des caractéristiques locales :

$$x_n = \mathbf{W}^{conv}([\mathbf{C}]_{c_{n-n_c+1}} : .. : [\mathbf{C}]_{c_n})^T + \mathbf{b}^{conv}$$

où x_n est un vecteur de dimension d_r obtenu pour chaque position n du mot [1]. Calculer la moyenne des i-èmes éléments des vecteurs de caractéristiques obtenus, à laquelle on applique la fonction d'activation ϕ :

$$[\mathbf{r}_w^{char}]_i = \phi\left(\sum_{n=1}^{|w|-n_c+1} \frac{[\mathbf{x}_n]_i}{|w| - n_c + 1}\right) \tag{1}$$

nous permet d'obtenir la représentation du mot w, $\mathbf{r}_w^{char}$. L'utilisation de la moyenne sur les résultats d'une convolution permet d'assurer que la représentation finale du mot combinera les caractéristiques locales extraites de l'ensemble du mot, et chacune des dimensions du gradient obtenu pour la représentation du mot est redistribué à chaque fenêtre de caractères à l'échelle de leur contribution à celles-ci. Les paramètres de cette couche sont les matrices $\mathbf{C}$ et $\mathbf{W}^{conv}$ et le biais $\mathbf{b}^{conv}$. L'ensemble du procédé est illustré figure 1, dans l'encadré bleu.

2.2 Modèles

Notre modèle est calqué sur l'architecture n-gramme *feed-forward*. Il prend en entrée un contexte de n mots $H_i = (w_{i-1}, \ldots, w_{N-i+1})$, et estime la distribution de probabilité du mot cible suivant : l'ensemble des valeurs $P(w|H_i)$ pour les mots du vocabulaire $w \in \mathcal{V}$. Quelle que soit la manière de les calculer, les représentations continues des mots du contexte sont concaténées puis sont transformées par une couche cachée qui applique une fonction d'activation non-linéaire [2] :

$$\mathbf{h}^{H_i} = \phi(\mathbf{W}^{hidden}(\mathbf{r}_{i-1} : \ldots : \mathbf{r}_{N-i+1}) + \mathbf{b}^{hidden}).$$

Enfin, la couche de sortie permet de calculer le score de chaque mot :

$$\mathbf{s}^{H_i} = \exp\left(\mathbf{W}^{out}\mathbf{h}^{H_i} + \mathbf{b}^{out}\right) \tag{2}$$

L'application de la fonction d'activation softmax permet de normaliser ces scores et d'obtenir pour chaque w la probabilité d'être le mot cible :

$$P(w|H_i) = \frac{\exp \mathbf{s}_w^{H_i}}{\sum_{1<j<|\mathcal{V}|} \exp \mathbf{s}_j^{H_i}}$$

1. Deux caractères spéciaux marquant le début et la fin des mots sont utilisés pour garder un nombre de fenêtres égal à la longueur du mot. Ils ont leur propre représentation, apprise et stockée dans $\mathbf{C}$.
2. Il est possible d'ajouter une deuxième couche cachée.

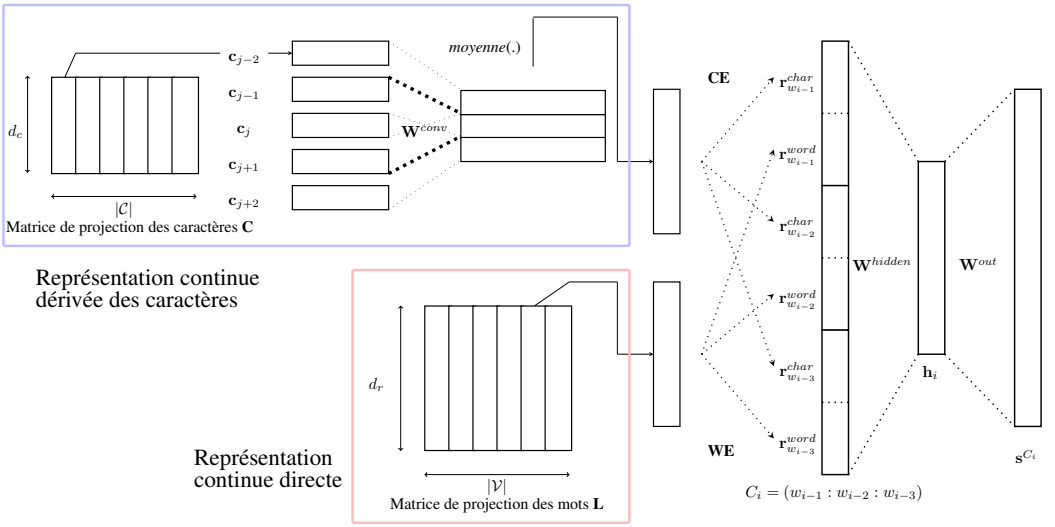

FIGURE 1: Architecture du modèle **CWE**

Les paramètres du modèle sont les matrices $\mathbf{W}^{hidden}$, $\mathbf{b}^{hidden}$, $\mathbf{W}^{out}$ et $\mathbf{b}^{out}$. De la même façon que la matrice $\mathbf{L}$, la matrice des poids de la couche de sortie $\mathbf{W}^{out}$ contient des représentations continues des mots du vocabulaire :

$$\mathbf{r}\text{-out}_w = [\mathbf{W}^{out}]_w$$

On peut déjà définir trois types de modèles :

— Un modèle n'utilisant que les représentations de mots, directement projetées dans un espace de dimension d_r - que l'on notera **WE**, pour *Word Embeddings*.

— Un modèle n'utilisant en entrée que les représentations dérivées des caractères, que l'on notera **CE**, pour *Character-level Embeddings*.

— Un modèle utilisant les deux types de représentations, concaténées, que l'on notera **CWE**, illustré figure 1.

2.3 Représentations dérivées des caractères pour la couche de sortie

Puisque la matrice de paramètres contenant les poids de sortie $\mathbf{W}^{out}$ peut être interprétée comme table des représentations continues de sortie des mots du vocabulaire, avec :

$$\mathbf{r}\text{-out}_w = [\mathbf{W}^{out}]_w$$

on peut choisir, comme pour le contexte donné en entrée, de calculer la représentation continue d'un mot à prédire à partir de ses caractères, à l'aide d'une couche de convolution.

Bien que nous n'introduisions pas de mécanisme de génération d'une séquence de caractères en sortie, nous donnons la possibilité au modèle de représenter et de calculer un score pour n'importe quel mot, même si il n'a pas été rencontré lors de l'apprentissage, là où auparavant tous les mots inconnus partageaient la même représentation, et donc le même score. Si, lors de l'apprentissage, l'on utilise un vocabulaire tiré des données d'apprentissage, on peut, lors de l'inférence, donner un

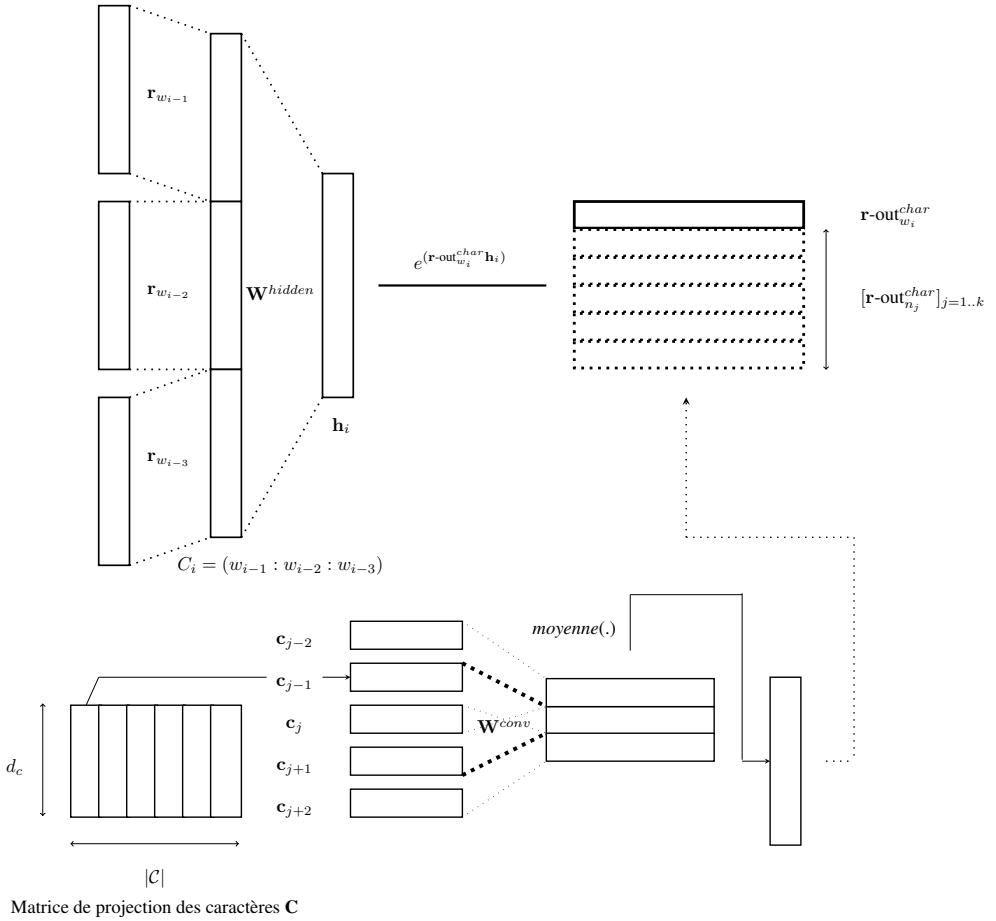

FIGURE 2: Architecture d'un modèle à vocabulaire ouvert en sortie

score à n'importe quel mot, ce qui est particulièrement utile dans le cadre de la ré-évaluation des N-meilleures hypothèses fournies par un système de traduction.

Ainsi, au lieu d'estimer le score comme dans l'équation 2, $\mathbf{r}\text{-out}_w$ peut être remplacé par un vecteur $\mathbf{r}\text{-out}_w^{char}$ estimé directement à partir de la séquence de ses caractères, de la même manière qu'à l'équation 1.

2.4 Fonction objectif pour les modèles à vocabulaire ouvert

La fonction objectif généralement utilisée pour l'entraînement de modèles de langue neuronaux consiste à maximiser la log-vraisemblance. Il s'agit d'estimer les paramètres θ qui permettent de maximiser la quantité suivante, sur l'ensemble des n-grammes (formés d'un mot cible w et de son

contexte H) observés dans les données d'apprentissage :

$$LL(\theta) = \sum_{1<i<|\mathcal{D}|} \log P_\theta(w_i|H_i)$$

ce qui correspond à la somme des log-probabiltiés conditionelles des mots, étant donné leur contexte.

Dans un modèle de langue conventionnel, le calcul de la distribution de probabilité conditionelle nécessite une normalisation sur l'ensemble des mots du vocabulaire $|\mathcal{V}|$, ce qui présente deux difficultés. D'abord, la normalisation implique le calcul d'une somme très couteûse, surtout pour les tailles de vocabulaire rencontrées en traduction automatique. Plus important encore, le calcul de cet objectif implique la définition d'un vocabulaire fini : l'extention décrite à la section 2.3 permet justement de ne pas en définir, pour pouvoir calculer à l'inférence des scores pour des mots qui n'ont pas été rencontrés pendant l'apprentissage.

Ainsi, on choisit d'utiliser l'estimation contrastive bruitée (NCE), introduite dans (Gutmann & Hyvärinen, 2012; Mnih & Teh, 2012), pour l'estimation des paramètres. L'idée du NCE est d'apprendre à distinguer les exemples provenant des données d'exemples négatifs échantillonnés à partir d'une distribution dite de bruit P_n. On cherche ainsi à décrire la distribution des données P_d relativement à cette distribution de bruit, qui sert de référence. On procède en tirant, pour chaque mot w et son contexte H provenant des données $\mathcal{D}$, k échantillons de bruit de la distribution P_n. Si C^w est la variable aléatoire binaire représentant l'appartenance d'un exemple w aux données ($C^w = 1$) ou au bruit (C^w = 0), les probabilités à priori pour les mots d'appartenir à ces classes sont :

$$P(C^w = 1) = \frac{1}{k+1} \text{ et } P(C^w = 0) = \frac{k}{k+1}$$

Et les probabilités conditionelles des mots pour ces ces classes sont :

$$P(w|H, C^w = 1) = P_\theta(w|H) \text{ et } P(w|H, C^w = 0) = P_n(w)$$

ce qui nous permet d'obtenir la probabilité *a posteriori* suivante pour l'appartenance aux données d'entraînement :

$$P^H(C^w = 1) = \frac{P_\theta(w|H)}{P_\theta(w|H) + kP_n(w)} \tag{3}$$

Le modèle $P_\theta(w|H)$ peut alors se paramétriser en considérant le score de sortie assigné à un mot w dans son contexte H :

$$s_\theta^H(w) = \exp\left(\mathbf{r}\text{-out}_w \mathbf{h}^H + \mathbf{b}_w\right). \tag{4}$$

Ce score peut ensuite être normalisé en introduisant le paramètre c^H propre à chaque contexte : $P_\theta(w|H) = s_\theta^H(w)\exp(c^H)$. Cependant, cette dépendance au contexte rend l'intégration de ce nouveau paramètre peu pratique. Dans (Mnih & Teh, 2012), les auteurs choisissent de fixer ces paramètres à zéro, le nombre de paramètres libres du modèle étant supposé suffisant pour que les scores non-normalisés de sortie s_θ^H s'auto-normalisent durant l'apprentissage ou du moins converge vers une somme constante pour chaque contexte. Comme suggéré dans (Mnih & Teh, 2012), P_n ne dépend pas du contexte, puisque l'on choisit pour bruit la distribution unigramme estimée sur les données d'apprentissage.

La fonction objectif finale s'obtient en maximisant la log-vraisemblance pour l'exemple des données w de vérifier $C^w = 1$ et pour les échantillons de bruit $(w_j^n)_{1 \leq j \leq k}$ de vérifier $C^w = 0$. Elle prend donc la forme suivante :

$$J_\theta^H = E_{s_\theta^H}\left[log \frac{s_\theta^H(w)}{s^H(w) + kP_n(w)}\right] + kE_{P_n}\left[log \frac{kP_n(w)}{s_\theta^H(w) + kP_n(w)}\right],$$

Il est possible de montrer (Gutmann & Hyvärinen, 2012) que lorsque k tend vers l'infini, cet objectif tend vers la log-vraisemblance $LL(\theta)$. Cette méthode de calcul approché permet donc de maximiser la log-vraisemblance d'un modèle en l'évaluant sur $k + 1$ exemples, au lieu de l'ensemble des mots d'un vocabulaire fixé.

Ainsi, cet objectif a l'avantage de fonctionner pour les différents types de modèles que l'on cherche à entraîner, qu'ils utilisent un vocabulaire fini ou non.

3 Cadre expérimental

La mesure traditionelle de performance pour les modèles de langue est la perplexité :

$$PPL = \exp\left(\sum_{j=1}^{|\mathcal{D}|} \frac{-\log P(w_j|H_j)}{|\mathcal{D}|}\right)$$

Cependant, d'une part, l'objectif principal de notre travail est la traduction automatique. D'autre part, la perplexité repose sur la définition d'un vocabulaire. Afin de comparer les différentes architectures présentées, on choisit donc de se servir de nos modèles neuronaux pour trier les sorties fournies par un système de traduction traditionel. Il s'agit donc de ré-évaluer les N meilleures hypothèses (N-*best reranking*), sur la tâche de traduction automatique de la campagne d'évaluation IWSLT16, de l'Anglais vers le Tchèque.

3.1 Description des données

La tâche de traduction automatique IWSLT16 se focalise sur la traduction de la transcription des séminaires TED Talks. Les systèmes de traduction sont entraînés sur les données parallèles des corpus **TED**, **QED** et **europarl**, respectivement constitués de $117K$, $125K$ et $642K$ phrases, ainsi que sur des données monolingues (**news** de 2007 à 2015, **news-commentary** et **europarl**), pour un total de $89M$ de phrases. Nos modèles de langue neuronaux sont entraînés sur les mêmes données, mais les exemples d'entraînement sont échantillonnés à partir de ces corpus suivant des coefficients calculés de manière à équilibrer les données parallèles qui sont dans ou hors du domaine, ainsi que les données monolingues additionelles. Enfin, on utilise la concaténation de **TED.dev2010**, **TED.dev2011** et de **TED.tst2010** comme données de validation, puis de **TED.tst2012** et de **TED.tst2013** comme données de test.

3.2 Système de traduction

Nous utilisons, pour notre système de référence, un système de traduction basé sur des n-grammes bilingues, NCODE [3], décrit dans (Crego *et al.*, 2011) et la configuration reprend celle décrite par (Marie *et al.*, 2015).

3.3 Configuration des modèles de langue neuronaux

En ce qui concerne l'apprentissage des modèles de langues, nous avons premièrement effectué des expériences comparatives à petite échelle, pour mieux comprendre le comportement des modèles à vocabulaire ouvert. Nous avons d'abord appris nos modèles avec une descente de gradient stochastique, mais l'instabilité de l'apprentissage restreignait le choix d'hyperparamètres : en effet, des dimensions de représentations des mots trop grandes, ainsi que certaines fonctions d'activation, rendaient la convergence de la fonction objectif du NCE erratique. Ces difficultées furent renforcées en travaillant avec le Tchèque, que nous avons trouvé plus difficile à manipuler que d'autres langues morphologiquement complexes, comme l'Allemand ou le Russe. L'utilisation d'Adagrad (Duchi *et al.*, 2010) nous a cependant permis de résoudre la plupart de ces problèmes.

Suite à des résultats préliminaires obtenus en travaillant sur des modèles similaires, mais pour une tâche différente (Labeau *et al.*, 2015), nous avons fait le choix de ne pas utiliser de réseaux récurrents pour obtenir des représentations continues des mots dérivées des caractères. Les résultats que nous avions obtenus étaient en effet similaires à ceux donnés par la couche de convolution, pour un temps de calcul accru.

L'apprentissage des modèles utilise des paquets (*batchs*) de taille 128 pour les trois types de modèles et les différentes tailles de contexte : d'abord, des modèles **WE** et **CWE**, puis des modèles utilisant à la fois en entrée et en sortie les deux types de représentations, suivant la structure présentée section 2.3, que l'on notera **CWE-CWE**.

À la suite des expériences préliminaires, nous avons choisi la fonction d'activation ReLu (Krizhevsky *et al.*, 2012), et des dimensions pour les représentations continues des mots et caractères de $d_r = 128$ et de $d_c = 32$. [4] La couche de convolution s'applique par fenêtres de $n_c = 5$ caractères et 25 exemples négatifs sont échantillonnés de la distribution de bruit pour chaque exemple positif provenant des données.

3.4 Ré-évaluation des N-meilleures hypothèses

L'étape de ré-évaluation et de tri des hypothèses utilise des scores additionnels pour choisir la meilleure traduction parmis les N meilleures hypothèses générées par le décodeur ($N = 300$). L'étape du *tuning* est réalisée avec KB-MIRA (Cherry & Foster, 2012), et on évalue les résultats à l'aide du score BLEU (Papineni *et al.*, 2002).

3. http ://ncode.limsi.fr

4. Les résultats n'ont pas changé de manière significative en augmentant la taille de ces représentations, alors que les temps de calcul et de convergences étaient accrus.

4 Résultats expérimentaux

La première série d'expériences cherche à déterminer l'impact des caractères spéciaux ajoutés en début et fin de mots pour faciliter l'application de convolution sur des fenêtres de caractère. Nous étudions ensuite le comportement des modèles et de la fonction objectif durant l'apprentissage. Enfin, nous évaluons nos modèles dans le contexte de la traduction automatique. Pour finir, nous abordons le problème de performance rencontré avec les modèles utilisant des représentations basées sur les caractères en sortie.

4.1 Impact des interdépendances entre les représentations dérivées de caractères en sortie

Dériver les représentations des mots à partir de leurs caractères implique que les représentations apprises soient liées - car les vecteurs représentant les caractères sont partagés. C'est une propriété importante, permettant au modèle de généraliser aux mots inconnus ce qu'il a appris auparavant. Dans notre cas, cela peut poser problème : notre fonction objectif ne permet qu'un nombre limité de mises à jour ($k + 1$), et elle utilise une distribution de bruit unigramme. Ainsi, les représentations de sortie mises à jour le plus fréquemment sont forcément celles des mots les plus fréquents. Ceux-ci sont des mots très courts, contenant peu de caractères, et les suites de n_c caractères les constituant se retrouveront facilement dans un grand nombre d'autres mots. Cela revient à dire qu'une grande partie du vocabulaire sera indirectement affectée par des mises à jour fréquentes de quelques mots très souvent échantillonnés. Les résultats préliminaires avec des représentations à base de caractères se sont avérés insatisfaisants, et nous avons supposé que les mises à jour trop fréquentes des mots courts en étaient la raison.

En effet, l'utilisation des caractères spéciaux (*padding*) en début et fin des mots, comme présentée dans la partie gauche de la table 1, que nous avions voulu simple, implique que des mots commençant ou finissant de la même façon partagent une ou plusieurs fenêtres de n_c caractères, ce qui rendra leur représentation finale très dépendantes. Nous avions d'abord supposé que cette inter-dépendance faciliterait la détection de similarités entre mots partageant préfixes et/ou suffixes. Cependant, la distribution des mots tchèques fait qu'un très petit nombre de mots affecte un très grand nombre de formes, puisque plus d'un tiers des données d'entraînement partage son commencement avec l'un des dix mots les plus fréquents, et plus d'un quart sa terminaison.

L'idée de considérer plusieurs tailles de fenêtres de convolution, comme dans (Kim *et al.*, 2015), pourrait aider à résoudre ce problème, mais cela augmenterait de manière drastique le temps de calcul de la couche de convolution. Nous avons préféré modifier la méthode de *padding* en n'ajoutant que le nombre de caractères spéciaux strictement nécessaire à l'application d'au moins une convolution au début et à la fin des mots, comme illustré dans la partie à droite de la table 1.

4.2 Apprentissage des modèles de langue neuronaux

Bien que l'objectif premier de ce travail ne soit pas d'améliorer la perplexité de notre modèle, elle est toujours liée à la quantité que notre objectif optimise - puisque le gradient du NCE est une approximation de celui du maximum de vraisemblance (Mnih & Teh, 2012).

○○○○a●●●●		○○a●●	
○○○○a l e●●●●	○○○○n a●●●●	○a l e●	○○n a●
○○○○a b y●●●●	○○○○z a●●●●	○a b y●	○○z a●
○○○○a ž●●●●	○○○○b y l a●●●●	○○a ž●	○b y l a●
○○○○a n i●●●●	○○○○d v a●●●●	○a n i●	○d v a●
○○○○a s i●●●●	○○○○t ř e b a●●●●	○a s i●	○t ř e b a●

TABLE 1: Ajout de caractères spéciaux pour la décomposition des mots en fenêtres de $n_c = 5$ caractères : ○ indique le début d'un mot, et ● la fin. A gauche, notre première façon d'ajouter ces caractères implique qu'un grand nombre de mots très différents aient des fenêtres de 5 caractères en commun, particulièrement avec des mots courts et fréquents. A droite, les modifications apportées.

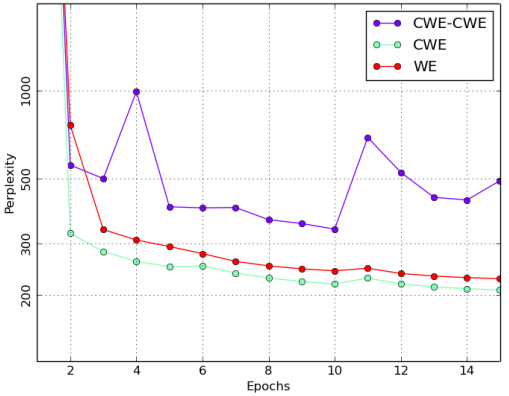

FIGURE 3

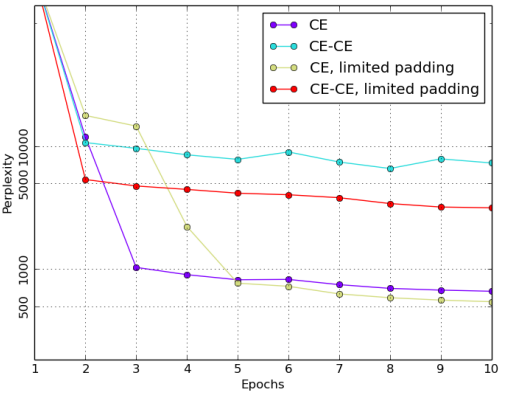

FIGURE 4

FIGURE 5: Perplexité des modèles, mesurée sur l'ensemble de validation, pendant l'entraînement. La taille du contexte est de 3 mots. La figure 4 montre des modèles entrainés avec les deux façons d'ajouter des caractères spéciaux présentées en section 4.1 : avec ou sans *padding* complet.

Les figures 3 et 4 donnent les courbes des perplexités de différents modèles pendant l'entraînement. Ces valeurs sont calculées sur un vocabulaire contenant les 250K mots les plus fréquents des données d'apprentissage. C'est aussi le vocabulaire utilisé par les modèles quand cela est pertinent. Le calcul de ces valeurs est réalisé sur un ensemble de validation après chaque époque. Une époque inclus 2,5M n-grammes, échantillonnés à partir des données d'entraînement. Le tableau 2 contient les meilleures perplexités obtenues sur l'ensemble de validation par chaque modèle, pendant l'apprentissage.

Le tableau 2 montre que les représentations dérivées des caractères aident à réduire la perplexité, même si augmenter la taille du contexte réduit l'écart. Les modèles utilisant une couche de convolution en sortie ont une perplexité un peu plus haute, même si encore une fois la taille permet de réduire l'écart. Il est cependant difficile d'interpréter la perplexité sur un modèle lorsqu'elle n'est calculée que sur une portion des mots que le modèle a scoré.

Le principal inconvénient d'Adagrad est que le taux d'apprentissage, déterminé par l'accumulation de l'historique des gradients précédents, est souvent trop aggressif, ce qui mène à un arrêt prématuré

Taille du contexte (en mots)	3	6
WE	227	193
CWE	207	185
CWE-CWE	308	243

TABLE 2: Perplexité obtenue sur l'ensemble de validation, calculée sur un vocabulaire de 250K mots, après 15 époques de 2,5M de n-grammes.

	Score BLEU de référence	CWE		CWE-CWE		WE	
		n=3	n=6	n=3	n=6	n=3	n=6
En → Cz	19.6	20.1	**20.3**	19.8	20.0	20.0	20.2

TABLE 3: Meilleurs scores BLEU obtenus après la ré-évaluation des 300 meilleures hypothèses données par le système de traduction. n est la taille du contexte (en nombre de mots)

de l'apprentissage. Un remède simple consiste à remettre à zéro cet historique toutes les 5 époques, pour laisser au modèle une possibilité d'amélioration - ce qui explique les allures des courbes de la figure3. Cependant, en dépit de l'utilisation d'un gradient adaptatif, l'entraînement d'un modèle avec une couche de convolution en sortie reste instable.

4.3 Re-évaluation des N-meilleures hypothèses

Les résultats de la ré-évaluation des hypothèses sont présentés dans le tableau 3. Le meilleur résultat est donné par le modèle **CWE** : avec une amélioration de $+\mathbf{0.7}$ points BLEU, ce qui est en moyenne 0.1 points BLEU de plus que le modèle **WE**. Le modèle **CWE-CWE** atteint des scores inférieurs (-0.2 points BLEU). De manière consistante, doubler la taille du contexte a augmenté le score de $+0.2$ points BLEU.

Les résultats obtenus par les modèles à vocabulaire de sortie ouverts sont en partie décevants, puisqu'ils sont supposés se révèler plus efficaces pour les mots inconnus. Nous conjecturons que c'est l'entraînement - et notamment les exemples présentés au modèle, qui posent problème avec en premier lieu leur manque de diversité : en effet, les exemples négatifs fournis par le NCE viennent d'un vocabulaire fermé. Le problème peut aussi venir de la nature de la distribution unigramme, utilisée pour échantillonner les exemples négatifs. Comme expliqué dans la section 4.1, les mots courts et plus fréquents l'emportent sur les autres en nombre de mises à jour, ce qui nous force à réduire la capacité du modèle à trouver des attributs morphologiques communs entre les mots.

5 Travaux liés

Il existe de nombreuses stratégies pour l'entraînement de modèles de langue neuronaux à grands vocabulaires, comme différentes couches de sortie hiérarchiques (Mnih & Hinton, 2009; Le *et al.*, 2011), ou approches liées à l'échantillonage (*Importance Sampling* (Bengio & Sénécal, 2003), et *Noise contrastive estimation* (Gutmann & Hyvärinen, 2012; Mnih & Teh, 2012)) (Vaswani *et al.*, 2013) a montré l'intérêt d'entraîner un modèle de langue avec le NCE pour la ré-évaluation des

n-meilleures hypothèses, tandis que (Devlin *et al.*, 2014) utilises un objectif modifié permettant l'auto-normalisation. Récemment, une étude comparative (Chen *et al.*, 2016) a été réalisée sur les solutions liées aux grands vocabulaires. Cependant, le but de ce travail est l'exploration de modèles à vocabulaire ouvert plutôt que les vocabulaires étendus.

L'extraction d'information au niveau des caractères a été récemment beaucoup explorée, permettant l'amélioration des résultats pour des tâches comme l'étiquetage en partie du discours (Santos & Zadrozny, 2014; Labeau *et al.*, 2015), la classification de textes (Zhang & LeCun, 2015), l'analyse syntaxique (Ballesteros *et al.*, 2015), et la reconnaissance d'entitées nommées (Lample *et al.*, 2016).

En ce qui concerne les modèles de langues, les premières applications fonctionnaient strictement au niveau du caractère, parvenant à de moins bons résultats que les modèles entrainés avec des mots (Mikolov *et al.*, 2012), bien que les résultats pour la génération de textes avec des LSTM bi-directionnels soient prometteurs (Sutskever *et al.*, 2011; Graves, 2013). Plus récemment, (Ling *et al.*, 2015) a utilisé des LSTM bi-directionnels pour construire des représentations continues de mots à partir de caractères, permettant d'améliorer les résultats de modèles de langues et d'étiquetage de séquences.

Les travaux présentés dans (Kim *et al.*, 2015), utilisant des couches de convolution pour construire des représentations de mots à partir de séquences de caractères, couplé à l'utilisation de *highway networks* (Srivastava *et al.*, 2015), ont démontré que les représentations continues dérivées des séquences de caractères permettent d'améliorer les performances des modèles de langues, et d'autant plus pour les langues morphologiquement complexes (sur des corpus de taille raisonnable). Une architecture similaire est présentée par (Józefowicz *et al.*, 2016) sur des données bien plus conséquentes, avec des LSTM bi-directionnels, et entraînés par *importance sampling*.

Sur le sujet de l'application de modèles de langues neuronaux à la traduction automatique, on peut mentionner le travail de (Luong *et al.*, 2015), sur l'influence du nombre de couches cachées sur la performance de la ré-évaluation des hypothèses d'un système de traduction. Enfin, bien que cela ne soit pas directement relié à nos travaux, (Luong & Manning, 2016) a obtenu des progrès significatifs à l'aide d'un système de traduction neuronal hybride, obtenant des représentations des mots inconnus à l'aide de leurs caractères.

6 Conclusion

Dans cet article, nous avons présenté un modèle de langue neuronal fonctionnant sans vocabulaire fixe. Les représentations continues des mots sont calculées à la volée à partir de la séquence de leurs caractères. Nous avons entrainé deux nouveaux types de modèles : d'abord, des modèles utilisant les deux types de représentations en entrée, pour représenter le contexte du mot à prédire (**CWE**), puis en étendant ces modèles afin qu'ils puissent prendre en comptes les représentations dérivées des caractères également en sortie pour la prédiction (**CWE-CWE**).

Ces modèles ont été utilisés pour la ré-évaluation d'hypothèses fournies par un système de traduction automatique de l'Anglais vers le Tchèque. Les meilleures améliorations obtenues utilisent un modèle **CWE**, ce qui, étant donné la diversité limitée des mots générés par le système de traduction, nous fait supposer qu'il reste une marge de progression. Nous espérons reproduire nos expériences avec des systèmes de traduction plus élaborés, ainsi qu'appliquer nos modèles à d'autres tâche - comme la ré-inflexion.

Nos expériences ont également montré que l'introduction d'un vocabulaire ouvert en sortie d'un modèle neuronal de langue entraîne des difficultés en terme d'apprentissage qu'il serait profitable d'explorer plus en profondeur. En particulier, nous prévoyons d'explorer des distributions de bruit plus évoluées permettant lors de l'apprentissage du modèle de présenter au modèles des exemples plus diversifiés.

Remerciements

Ces travaux ont été en partie financés par le programme Européen pour la recherche et l'innovation Horizon 2020, à travers la subvention No. 645452 (QT21), ainsi que l'Agence Nationale de la Recherche (projet ParSiTi, ANR-16-CE33-0021). Nous remercions les relecteurs pour leurs commentaires et suggestions.

Références

ALLAUZEN A. & GAUVAIN J. (2005). Open vocabulary asr for audiovisual document indexation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

BAHDANAU D., CHO K. & BENGIO Y. (2014). Neural machine translation by jointly learning to align and translate. *CoRR*, **abs/1409.0473**.

BALLESTEROS M., DYER C. & SMITH N. A. (2015). Improved transition-based parsing by modeling characters instead of words with lstms. In *EMNLP*, p. 349–359.

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

BENGIO Y. & SÉNÉCAL J.-S. (2003). Quick training of probabilistic neural nets by importance sampling. In *Proceedings of the conference on Artificial Intelligence and Statistics (AISTATS)*.

BOTHA J. A. & BLUNSOM P. (2014). Compositional Morphology for Word Representations and Language Modelling. In *Proceedings of the International Conference of Machine Learning (ICML)*.

CHEN W., GRANGIER D. & AULI M. (2016). Strategies for training large vocabulary neural language models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016*.

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

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

CREGO J. M., YVON F. & MARIÑO J. B. (2011). N-code : an open-source Bilingual N-gram SMT Toolkit. *Prague Bulletin of Mathematical Linguistics*, **96**, 49–58.

DEVLIN J., ZBIB R., HUANG Z., LAMAR T., SCHWARTZ R. M. & MAKHOUL J. (2014). Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014*, p. 1370–1380.

DUCHI J., HAZAN E. & SINGER Y. (2010). *Adaptive Subgradient Methods for Online Learning and Stochastic Optimization*. Rapport interne.

GRAVES A. (2013). Generating sequences with recurrent neural networks. *CoRR*, **abs/1308.0850**.

GUTMANN M. U. & HYVÄRINEN A. (2012). Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *J. Mach. Learn. Res.*, **13**(1).

JÓZEFOWICZ R., VINYALS O., SCHUSTER M., SHAZEER N. & WU Y. (2016). Exploring the limits of language modeling. *CoRR*, **abs/1602.02410**.

KIM Y., JERNITE Y., SONTAG D. & RUSH A. M. (2015). Character-aware neural language models. *arXiv preprint arXiv :1508.06615*.

KRIZHEVSKY A., SUTSKEVER I. & HINTON G. E. (2012). Imagenet classification with deep convolutional neural networks. In F. PEREIRA, C. J. C. BURGES, L. BOTTOU & K. Q. WEINBERGER, Eds., *Advances in Neural Information Processing Systems 25*, p. 1097–1105.

LABEAU M., LÖSER K. & ALLAUZEN A. (2015). Non-lexical neural architecture for fine-grained pos tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, p. 232–237, Lisbon, Portugal.

LAMPLE G., BALLESTEROS M., KAWAKAMI K., SUBRAMANIAN S. & DYER C. (2016). Neural architectures for named entity recognition. In *In proceedings of NAACL-HLT (NAACL 2016)*.

LE H.-S., ALLAUZEN A. & YVON F. (2012). 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.

LE H. S., OPARIN I., ALLAUZEN A., GAUVAIN J. & YVON F. (2011). Structured output layer neural network language model. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2011*, p. 5524–5527.

LING W., DYER C., BLACK A. W., TRANCOSO I., FERMANDEZ R., AMIR S., MARUJO L. & LUÍS T. (2015). Finding function in form : Compositional character models for open vocabulary word representation. In *EMNLP*, p. 1520–1530 : The Association for Computational Linguistics.

LUONG M. & MANNING C. D. (2016). Achieving open vocabulary neural machine translation with hybrid word-character models. *CoRR*, **abs/1604.00788**.

LUONG T., KAYSER M. & MANNING C. D. (2015). Deep neural language models for machine translation. In *CoNLL 2015, Beijing, China, July 30-31, 2015*, p. 305–309.

MARIE B., ALLAUZEN A., BURLOT F., DO Q.-K., IVE J., KNYAZEVA E., LABEAU M., LAVERGNE T., LÖSER K., PÉCHEUX N. & YVON F. (2015). Limsi@wmt'15 : Translation task. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, p. 145–151.

MIKOLOV T., SUTSKEVER I., DEORAS A., LE H.-S., KOMBRINK S. & CERNOCKY J. (2012). Subword language modeling with neural networks. Unpublished.

MNIH A. & HINTON G. (2009). A scalable hierarchical distributed language model. In *Advances in Neural Information Processing Systems*, volume 21, p. 1081–1088.

MNIH A. & TEH Y. W. (2012). A fast and simple algorithm for training neural probabilistic language models. In *ICML* : icml.cc / Omnipress.

PAPINENI K., ROUKOS S., WARD T. & ZHU W.-J. (2002). Bleu : A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, p. 311–318.

SANTOS C. D. & ZADROZNY B. (2014). Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)* : JMLR Workshop and Conference Proceedings.

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

SENNRICH R., HADDOW B. & BIRCH A. (2016). Neural machine translation of rare words with subword units. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL)*, p. 1715–1725.

SRIVASTAVA R. K., GREFF K. & SCHMIDHUBER J. (2015). Training very deep networks. *CoRR*, **abs/1507.06228**.

SUTSKEVER I., MARTENS J. & HINTON G. (2011). Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, p. 1017–1024.

VASWANI A., ZHAO Y., FOSSUM V. & CHIANG D. (2013). Decoding with large-scale neural language models improves translation. In *EMNLP*, p. 1387–1392 : ACL.

WAIBEL A., HANAZAWA T., HINTON G., SHIKANO K. & LANG K. J. (1990). *Readings in Speech Recognition*, chapter Phoneme Recognition Using Time-delay Neural Networks.

ZHANG X. & LECUN Y. (2015). Text understanding from scratch. *CoRR*, **abs/1502.01710**.