Apprentissage non supervisé de représentations de mots à l'aide de réseaux de convolution bilinéaires sur des caractères

Thomas Luka, Laure Soulier, David Picard

Apprentissage non supervisé de représentations de mots à l'aide de réseaux de convolution bilinéaires sur des caractères

Thomas Luka* — **Laure Soulier**** — **David Picard***

* *LIGM, Ecole des Ponts, Univ Gustave Eiffel, CNRS*
Marne-la-Vallée, France
Email: {thomas.luka,david.picard}@enpc.fr
** *Sorbonne Université, CNRS, LIP6*
F-75005 Paris, France
Email: {thomas.luka,david.picard}@enpc.fr, laure.soulier@lip6.fr

RÉSUMÉ. *Dans cet article, nous proposons une nouvelle méthode non-supervisée pour apprendre des représentations de mots avec des convolutions directement sur des caractères. Nous évitons ainsi les problèmes inhérents à l'utilisation d'un dictionnaire. Pour y parvenir, nous avons traduit l'hypothèse de distribution par une fonction de coût d'apprentissage de métrique. Cela permet d'avoir un unique encodeur au lieu des architectures comportant un encodeur et un décodeur. Enfin, nous proposons d'utiliser un réseau convolutif comportant des connexions résiduelles et des produits bilinéaires pour être en mesure d'encoder des motifs de co-occurences. Nous démontrons l'efficacité de notre approche en la comparant avec les méthodes classiques comme fastText et GloVe sur différents jeux de données.*

ABSTRACT. *In this paper, we propose a new unsupervised method for learning word embedding with raw characters as input representation, bypassing the problems arising from the use of a dictionary. To achieve this purpose, we translate the distributional hypothesis into a unsupervised metric learning objective, which allows us to consider only an encoder instead of an encoder-decoder arhitecture. We propose to use a convolutional neural network with bilinear product blocks and residual connections to encode co-occurences patterns. We show the efficiency of our approach by comparing it with classical word embedding methods such as fastText and GloVe on several benchmarks.*

MOTS-CLÉS : *Représentations de mots, Produit bilinéaire, Apprentissage de métrique, Convolutions sur des caractères.*

KEYWORDS: *Word representations, Bilinear Pooling, Metric learning, Character convolutions*

1. Introduction

L'apprentissage de représentations de mots est une tâche importante pour déduire la sémantique des mots et des textes et est à la base de nombreuses méthodes abordant une grande variété de tâches, notamment la traduction automatique (Bahdanau *et al.*, 2015), les systèmes question-réponse (Bordes *et al.*, 2014) ou l'analyse de sentiments (Maas *et al.*, 2011). Nous nous intéressons dans cet article à la sémantique des mots et distinguons deux types de méthodes récentes d'apprentissage de représentations de mots : celles non contextualisées (un mot a une seule représentation quelque soit le contexte dans lequel il apparaît) et celles contextualisées (un mot a des représentations multiples qui dépendent du contexte dans lequel il apparaît).

Les méthodes d'apprentissage non contextualisées utilisent soit un dictionnaire des mots les plus fréquents (Mikolov *et al.*, 2013a ; Pennington *et al.*, 2014), soit des « sous-mots » (Bojanowski *et al.*, 2017), soit des convolutions sur des caractères (Kim *et al.*, 2016). Dans les approches avec dictionnaire (Mikolov *et al.*, 2013a ; Pennington *et al.*, 2014), la représentation d'entrée pour un mot est l'encodage 1 parmi n de son index dans le dictionnaire et cette représentation est ensuite traitée pour déduire la sémantique des mots et obtenir leur représentation dans l'espace sémantique. Ces méthodes présentent plusieurs inconvénients : les mots qui ne sont pas dans le dictionnaire, comme les mots rares et les mots mal orthographiés, sont tous associés à la même entrée réservée aux mots inconnus et ont donc la même représentation alors qu'ils ne partagent pas la même signification. En outre, ces méthodes nécessitent généralement l'entraînement d'un décodeur, inutilisé après l'entraînement, pour prédire les mots environnants.

Pour résoudre certains de ces inconvénients, des travaux comme (Bojanowski *et al.*, 2017), utilisent des n-grammes, c'est-à-dire des « sous-mots » de n caractères, pour représenter les mots. Un mot est alors la somme de ses n-grammes. Dans le domaine de la traduction, d'autres travaux (Sennrich *et al.*, 2016) ont proposé le *Byte-Pair-Encoding* pour pouvoir représenter les mots absents du dictionnaire. Un autre type de méthodes est basé sur des réseaux qui effectuent des convolutions directement sur des caractères bruts (Kim *et al.*, 2016). Ces méthodes permettent de résoudre certains problèmes inhérents aux méthodes basées sur les dictionnaires : les fautes de frappe et les mots hors vocabulaire, mais nécessite une fois de plus une étape coûteuse de décodage pour l'entraînement.

Les approches plus récentes se concentrent sur la représentation contextualisée des mots, comme dans (Peters *et al.*, 2018), avec l'inconvénient majeur qu'un mot ne peut pas avoir de représentation sans son contexte. Nous nous concentrons dans ce travail sur les représentations non contextualisées des mots.

Dans cet article, nous proposons BiCharaConv (**Bi**linear Pooling on **Chara**cter **Conv**olution), une méthode de convolution sur des caractères qui résout les problèmes découlant de l'utilisation d'un dictionnaire. Nous inspirant de la vision par ordinateur où la mise en œuvre des produits bilinéaires a montré des améliorations significatives des performances dans plusieurs tâches, notamment la reconnaissance à grain fin (Lin

et al., 2015 ; Gao *et al.*, 2016 ; Kong et Fowlkes, 2017) et la recherche multi-modale (Kim *et al.*, 2017), nous avons intégré dans notre architecture des opérations de ce type. L'hypothèse est que les produits bilinéaires peuvent nous être utiles dans une architecture convolutive pour capturer des co-occurrences entre les caractères et ainsi repérer les préfixes (in-,intra-,para-, bi-...), suffixes (-tion, -eur.e, -ment,...), etc. En outre, pour éviter une étape de décodage, nous reformulons l'*hypothèse de distribution* (Harris, 1954) en une fonction de coût d'apprentissage de métrique. Nous montrons l'efficacité de notre approche sur plusieurs jeux de données et montrons dans des études d'ablation la pertinence de notre méthode.

Dans la suite de cet article, nous présentons l'état de l'art des modèles d'apprentissage de représentation de mots (Section 2). Notre modèle, **Bi**linear Pooling on **Chara**cter **Conv**olution, est présenté dans la Section 3. Le protocole expérimental est détaillé dans la Section 4. Les résultats ainsi qu'une étude d'ablation sont respectivement présentés dans les Sections 5 et 6. Enfin, nous concluons le papier et présentons les perspectives de recherche en Section 7.

2. État de l'art

Dans cette section, nous faisons un bref état de l'art afin de positionner nos travaux par rapport aux différentes familles de méthodes d'apprentissage de représentation de mots de la littérature.

2.1. *Méthodes à dictionnaire*

En 2013, Mikolov *et al* introduisent dans (Mikolov *et al.*, 2013a) et (Mikolov *et al.*, 2013b) les méthodes bien connues de CBOW et skip-gram pour apprendre des représentations de mots. Skip-gram est une méthode à dictionnaire de N mots sous forme de vecteurs à encodage 1 parmi N projetés dans l'espace de représentation. Un décodeur tente ensuite de décoder les mots voisins. De la même manière, CBOW fait la même chose mais en inversé : le décodeur tente de décoder un mot à l'aide de ses voisins. Plus tard, Pennington *et al* introduisent GloVe (Pennington *et al.*, 2014), une approche similaire basée sur la factorisation de matrices.

Ces méthodes présentent l'inconvénient que tous les mots qui ne sont pas dans le dictionnaire comme les mots rares ou mal orthographiés ont la même représentation du « mot inconnu ». De plus, l'étape de décodage est coûteuse et nécessite l'apprentissage d'un décodeur qui devient inutile après l'entraînement.

2.2. *Stratégies pour représenter les mots rares ou inconnus*

Pour résoudre le problème introduit par l'utilisation d'un dictionnaire de mots et pour pouvoir représenter les mots rares, Wieting *et al* choisissent d'utiliser des n-

grammes, des « sous-mots » d'une longueur de n caractères, et introduisent le modèle CHARAGRAM (Wieting *et al.*, 2016) pour pouvoir représenter soit des mots soit des phrases. Une séquence de caractères est représentée comme la somme de ses n-grammes. Cette idée est également exploitée par Bojanowski *et al.* en 2017 (Bojanowski *et al.*, 2017) avec fastText. Dans leur travail, les n-grammes sont ensuite envoyés dans un dictionnaire de longueur fixe ($2 \cdot 10^6$) grâce à une fonction de hachage, puis convertis en vecteurs 1 parmi n. Ils suivent ensuite la méthode skip-gram pour l'entraînement et leur méthode présente donc les inconvénients de l'étape de décodage. Dans le domaine de la traduction, d'autres auteurs, inspirés par les algorithmes de compression, utilisent le *Byte-Pair-Encoding* (BPE) au niveau des caractères (Sennrich *et al.*, 2016) ou même au niveau des octets (Wang *et al.*, 2020) pour encoder les mots. WordPiece (Wu *et al.*, 2016), utilisé par BERT (Devlin *et al.*, 2018) pour représenter les mots, utilise le BPE au niveau du caractère. Une liste des sous-mots, de longueur variable, les plus fréquents est construite et un mot est alors une séquence de sous-mots. Cela présente l'avantage que les mots rares peuvent être représentés comme une séquence de sous-mots connus mais un mot n'est pas forcément représenté par un seul vecteur.

Dans (Zhang *et al.*, 2015), Zhang *et al* proposent l'utilisation de convolution 1D directement sur les caractères pour faire de la classification de textes et obtiennent des résultats encourageants. L'idée d'utiliser des convolutions est reprise un an plus tard dans (Kim *et al.*, 2016) par Kim *et al*. pour construire des représentations de mots avec des convolutions de caractères, des réseaux *highway* (Srivastava *et al.*, 2015) et un décodeur LSTM multi-couches. Cela nécessite un dictionnaire uniquement pour l'entraînement et permet de représenter les mots hors vocabulaire. Cependant, un décodeur est également formé et inusité après l'entraînement.

2.3. *Représentations de mots dépendantes du contexte*

À la frontière entre l'apprentissage de représentations de mots et de phrases, les recherches les plus récentes proposent d'inclure des informations contextuelles dans les représentations des mots afin d'améliorer les résultats des tâches utilisant des représentations de phrases notamment la réponse automatique aux questions ou l'analyse des sentiments. McCann *et al* (McCann *et al.*, 2017) proposent de concaténer les vecteurs de mots avec les dernières sorties d'un LSTM bidirectionnel formé à une tâche de traduction automatique. Peters *et al* (Peters *et al.*, 2018) vont plus loin et incluent des caractéristiques extraites grâce à des convolutions sur des caractères et tous les états intermédiaires d'un LSTM bidirectionnel à l'intérieur des représentations des mots dans leur modèle ELMo grâce à une somme pondérée. Plus récemment encore, des modèles basés sur les *transformers* (Vaswani *et al.*, 2017) qui mettent en œuvre un mécanisme d'auto-attention, comme BERT (Devlin *et al.*, 2018) et ses variantes, atteignent l'état de l'art sur une grande variété de tâches d'analyse de phrases comme en témoignent les performances de ces modèles sur GLUE (Wang *et al.*, 2018). Le

principal inconvénient de ces méthodes est qu'elles ne peuvent pas calculer la représentation d'un mot sans son contexte.

Dans ce travail, nous nous concentrons sur un encodeur de mot indépendant du contexte (à la différence d'ELMo (Peters *et al.*, 2018) ou de BERT (Devlin *et al.*, 2018)) et qui évite les limitations introduites par l'utilisation d'un dictionnaire et par le schéma d'entraînement utilisant des décodeurs grâce à de l'apprentissage de métrique non supervisé. Nous évaluons comment une architecture convolutive peut être compétitive lorsqu'elle implémente des produits bilinéaires permettant de détecter des préfixes, suffixes et autres motifs de caractères.

3. Méthode proposée

Dans cette section, nous présentons notre modèle BiCharaConv (**Bi**linear Pooling on **Chara**cter **Conv**olution). Nous aborderons dans un premier temps notre représentation d'entrée. Nous présenterons ensuite les motivations et l'implémentation des produits bilinéaires dans notre architecture neuronale. Nous conclurons avec notre procédure d'entraînement.

3.1. *Représentation d'un mot en entrée*

Nous avons choisi une matrice $X = (\mathbf{x}_j)_j \in \mathcal{M}_{l_w \times d_c}(\mathbb{R})$ comme représentation d'entrée pour un mot w. Chaque ligne $\mathbf{x}_j$, de dimension d_c, encode le $j^{ième}$ caractère d'un mot, c'est-à-dire une lettre de l'alphabet ou un caractère spécial (au nombre de deux : un pour *début/fin de mot* et un pour *caractère inconnu*). L'ensemble de vecteurs de représentation est choisi de sorte qu'ils soient orthonormaux. l_w est la longueur d'un mot : les mots plus longs sont tronqués et ceux plus courts sont répétés jusqu'à ce qu'ils correspondent à la longueur.

3.2. *Convolution sur des caractères avec produits bilinéaires pour détecter des motifs de caractères*

L'objectif de notre architecture est de combiner des co-occurrences en mêlant des logiques *OU* et des logiques *ET*. L'utilisation de co-occurrences du type *OU* permet d'encoder des options de motifs comme par exemple « il y a un 'a' *OU* un 'i' à la position p ». Notons que cela nous permet d'être robuste aux petites variations orthographiques. À l'inverse les co-occurrences de type *ET* permettent d'encoder des combinaisons de motifs, comme par exemple « il y a un 'a' à la position p ET un 'l' à la position $p + 1$ ». Cette logique permet de détecter des motifs plus précis et donc discriminants. La combinaison de ces deux logiques nous aide ainsi à détecter des motifs de caractères ou d'ensembles de caractères robustes aux erreurs typographiques, comme par exemple des combinaisons de préfixe, racine et suffixe et qui sont primordiaux pour la bonne compréhension des mots.

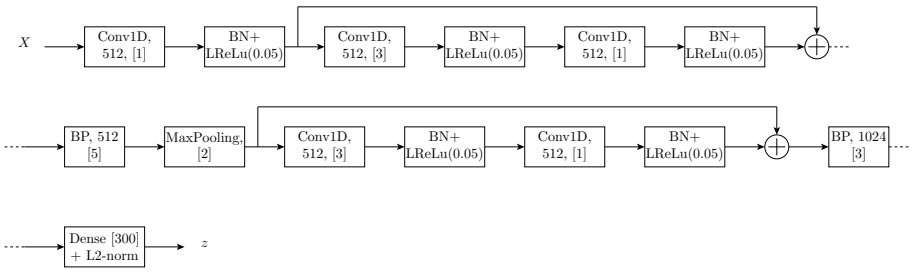

Figure 1 – BiCharaConv. X est l'entrée, z est la représentation du mot dans l'espace de représentation. $\mathrm{LReLu}(\alpha)$ correspond à une *Leaky ReLU* avec un paramètre α et les blocs BP implémentent le produit bilinéaire. Les quantités entres crochets indiquent la taille de voisinage de la fenêtre glissante.

Pour réaliser les co-occurrences de type *OU*, nous proposons d'utiliser des convolutions 1D (*Conv1D* suivi d'une *BatchNormalization* et d'une activation *LeakyReLU* (*BN+LReLu*)). Pour réaliser les co-occurrences du type *ET*, nous proposons d'utiliser des blocs de produit bilinéaire (*BP*) qui vont inspecter la matrice de covariance[1] de deux caractéristiques pour y détecter des directions représentant un motif résultant d'une combinaison particulière.

Plus formellement, considérons deux caractéristiques : une caractéristique $\mathbf{x_t}$, et une caractéristique décalée de k : $\mathbf{x_{t-k}}$. Nous calculons alors $\langle \mathbf{x_{t-k}} {\mathbf{x_t}}^T, \mathbf{W} \rangle$ avec $\mathbf{W}$ une matrice de projection apprise pour représenter un motif de co-occurrence de type ET.

Puisque le calcul direct est très coûteux (quadratique en nombre de dimension d'entrée), nous simplifions en utilisant les propriétés du produit scalaire et du produit de Kronecker en faisant l'hypothèse que $\mathbf{W}$ est de rang faible. Ainsi, on peut écrire :

$$\langle \mathbf{x_{t-k}} {\mathbf{x_t}}^T, \mathbf{W} \rangle = \left\langle \mathbf{x_{t-k}} {\mathbf{x_t}}^T, \sum_{i=1}^{r} \lambda_i \mathbf{u_i} {\mathbf{v_i}}^T \right\rangle \quad [1]$$

$$= \sum_{i=1}^{r} \lambda_i \langle \mathbf{x_{t-k}} \otimes \mathbf{x_t}, \mathbf{u_i} \otimes \mathbf{v_i} \rangle \quad [2]$$

$$= \sum_{i=1}^{r} \lambda_i \langle \mathbf{x_{t-k}}, \mathbf{u_i} \rangle \langle \mathbf{x_t}, \mathbf{v_i} \rangle \quad [3]$$

$$\langle \mathbf{x_{t-k}} {\mathbf{x_t}}^T, \mathbf{W} \rangle = \langle \Lambda, [\langle \mathbf{x_{t-k}}, \mathbf{u_i} \rangle \langle \mathbf{x_t}, \mathbf{v_i} \rangle]_i \rangle \ (\Lambda = [\lambda_i]_i) \quad [4]$$

1. En pratique, nous calculons la matrice d'ordre 2 pour éviter de centrer mais l'intuition reste la même.

Nous voulons intégrer toutes les caractéristiques dans un voisinage : nous proposons par conséquent d'implémenter nos blocs BP de produit bilinéaire comme la transformation définie ci-dessous :

$$\mathrm{BP}(\mathbf{x_t}) = \sum_k \langle \mathbf{x_{t-k}} \mathbf{x_t}^T, \mathbf{W_k} \rangle = \sum_k \langle \Lambda, [\langle \mathbf{x_{t-k}}, \mathbf{u_{i,k}} \rangle, \langle \mathbf{x_t}, \mathbf{v_i} \rangle]_i \rangle \quad [5]$$

L'équation 5 est facilement implémentable avec un produit terme à terme et des convolutions 1D (une convolution 1D avec un noyau égal à 1 est équivalent à un produit scalaire).

L'architecture globale est présentée sur la figure 1. Elle composée d'une première convolution 1D (suivie d'une *BatchNormalization* et d'une activation) de mise en forme du signal avant deux échelles d'analyses *OU-ET*. Chaque échelle comprend un bloc résiduel composé de deux convolutions 1D (plus *BatchNormalization* et activation) suivi d'un bloc de produit bilinéaire. La première échelle nous permet de détecter les préfixes, racines et suffixes tandis que la seconde échelle nous permet de détecter les combinaisons de motifs détectés par la première échelle pour construire des représentations de mot entier après une projection dans l'espace de représentation. Quant aux connections résiduelles, elles sont utilisées pour faciliter le processus d'apprentissage mais aussi pour que nos blocs de produit bilinéaire puissent avoir accès à une information de plus bas niveau au besoin (par exemple, retour aux caractères/motifs bruts au lieu d'options de caractères/motifs).

3.3. *Entraînement et orthogonalisation*

La méthode skipgram traduit l'*hypothèse de distribution* (Harris, 1954) à l'aide d'une architecture encodeur/décodeur. Cependant, l'étape de décodage a un coût de calcul élevé et le décodeur est inusité après l'entraînement. Nous proposons plutôt de placer directement dans l'espace de représentation un mot plus proche de son contexte grâce à une fonction de coût d'apprentissage de métrique. Commençons par introduire les notations suivantes :

– Soit $\mathcal{W}$ un ensemble de mots pris dans un corpus de phrases (potentiellement tronquées) de 11 mots chacune.

– $\forall w_q \in \mathcal{W}$, soit $\mathcal{P}(w_q) \subset \mathcal{W}$, l'ensemble des mots positifs qui contient les 10 mots voisins de w_q, *i.e.* les mots autour de w_q dans la même phrase.

– $\forall w_q \in \mathcal{W}$, soit $\mathcal{N}(w_q) \subset \mathcal{W}$ l'ensemble des mots négatifs, *i.e.* l'ensemble de tous les mots de $\mathcal{W}$ qui ne sont pas dans la même phrase que w_q.

– Soit $z = E_\theta(w)$ le vecteur représentant le mot w dans l'espace de représentation encodé par l'encodeur E qui a pour paramètres θ.

Nous proposons d'apprendre E_θ par apprentissage de métrique en utilisant la fonction de coût suivante :

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{w_q} \sum_{w_p \in \mathcal{P}(w_q)} \sum_{w_n \in \mathcal{N}(w_q)} \max\left\{0; ||z_q - z_p||^2 + \beta - ||z_q - z_n||^2\right\} \quad [6]$$

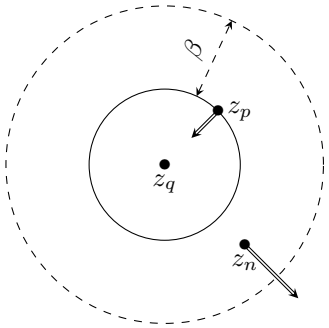

Figure 2 – Triplet loss

N est le nombre de triplets (w_q, w_p, w_n) actifs (ceux où $||z_q - z_p||^2 + \beta - ||z_q - z_n||^2 > 0$)
Cette fonction de coût, la triplet loss (Frome *et al.*, 2007), compare la distance entre un point de référence z_q et un point positif z_p- la représentation d'un mot du contexte de w_q - avec la distance entre le même point de référence et un point négatif z_n - la représentation d'un mot qui n'est pas dans la même phrase que w_q. La distance entre la référence et le négatif doit être au moins supérieur d'une marge β à la distance entre la référence et le positif. Une illustration est représentée sur la figure 2.

Par ailleurs, pour éviter l'effondrement de l'information utile sur quelques dimensions, les autres dimensions devenant du bruit pour les représentations, nous proposons d'ajouter la fonction de coût d'orthogonalisation ci-dessous :

$$\mathcal{L}_\perp = \left[\frac{1}{N} \left(\sum_{w \in \mathcal{B}} z z^T \right) - \frac{1}{N} \sum_{w \in \mathcal{B}} z \frac{1}{N} \sum_{w \in \mathcal{B}} z^T - I \right]^2 \quad [7]$$

où N est le nombre de mots dans un batch $\mathcal{B}$, $z = E_\theta(w)$ est la représentation du mot w dans l'espace de représentation, et I est la matrice identité. En fait, nous forçons l'estimation de la matrice de covariance du batch à tendre vers l'identité. Avec l'orthogonalisation, la fonction de coût globale $\mathcal{L}_{tot}$ devient :

$$\mathcal{L}_{tot} = \mathcal{L} + \gamma \mathcal{L}_\perp \quad [8]$$

avec γ un coefficient à ajuster et $\mathcal{L}$ la triplet loss.

4. Expérimentations

4.1. *Jeux de données pour l'apprentissage de représentation*

Dans ce travail, nous considérons quatre ensembles de données pour construire notre corpus de textes : les phrases de MS-COCO (Lin *et al.*, 2014), l'UMBC webbase

(Lushan Han, Abhay L. Kashyap, Tim Finin, James Mayfield, and Johnathan Weese, 2013), une sauvegarde des 50 premiers Go de Wikipédia et common-crawl.

Pour chaque ensemble de données, nous supprimons les phrases de moins de 10 mots afin de garantir un contexte approprié et de filtrer les éventuelles erreurs qui peuvent se produire lors des étapes de conversion html ou xml vers du texte brut ou lors des étapes de nettoyage. Toutes les phrases sont converties en minuscules.

4.2. *Tâches d'évaluation de la qualité de l'espace de représentation*

Pour évaluer la qualité de l'espace de représentation appris, nous utilisons la même configuration que dans (Zablocki *et al.*, 2018), c'est-à-dire plusieurs tâches qui reflètent la capacité de notre espace à prédire la concrétude, la similarité et la relation entre une paire de mots ainsi que les caractéristiques des objets. Dans ce qui suit, nous allons détailler chaque tâche et nos méthodes de référence.

4.2.1. *Concrétude*

L'évaluation du caractère concret est basée sur le score de concrétude de la base de données de l'USF (Nelson *et al.*, 2004). Les scores sont continus entre 1 et 7 et nous essayons de les prédire en faisant une SVR avec un noyau gaussien. Le score reporté est le coefficient de détermination $\mathcal{R}^2$ de la régression.

4.2.2. *Similarité et relation entre une paire de mots*

Le score de similarité et de relation entre deux mots est mesuré par la corrélation de Spearman entre la similarité cosinus d'une paire de mots dans l'espace de représentation et le score de similarité donné par les humains. Les jeux de données utilisés sont les suivants : WordSim353 (Finkelstein *et al.*, 2002), Visual-Sim et Semantic-Sim (Silberer et Lapata, 2014), USF (Nelson *et al.*, 2004), SimLex999 (Hill *et al.*, 2015), RW (Luong *et al.*, 2013), Mturk771 (Halawi *et al.*, 2012) et MEN (Bruni *et al.*, 2014). À noter que le jeu de données RW recense des paires de mots rares ou des mots composés contenant au moins un mot rare.

4.2.2.1. Prédiction de caractéristiques

Les données de McRae (McRae *et al.*, 2005) contiennent un ensemble de caractéristiques pour les concepts vivants et non vivants. Le test consiste à les prédire pour un mot grâce à une SVM linéaire. Par exemple, pour le mot *pomme*, nous voulons prédire *est rond*, *est comestible*, et ainsi de suite. Dans ces données, nous avons 417 entités et 43 caractéristiques à prédire regroupées en 9 connaissances. Les scores considérés sont la moyenne des F1-scores entre les prédictions et la vérité terrain.

4.3. *Méthodes de référence (Baselines)*

Dans ce travail, nous avons choisi les vecteurs de mots de **GloVe** (Pennington *et al.*, 2014) pré-entraînés sur 6 milliards de mots provenant d'un corpus composé de Wikipedia 2014 et Gigaword 5. Nous avons également choisi les vecteurs pré-entraînés de **fastText** (Bojanowski *et al.*, 2017 ; Mikolov *et al.*, 2018) pré-entrainés sur 600 milliards de mots provenant de Common-Crawl. Nous avons choisi ces deux méthodes car Glove est très populaire grâce à ses vecteurs de mots facilement accessibles ce qui rend son utilisation large et fastText utilise des n-grammes pour modéliser les mots rares.

4.4. *Détails d'implémentation*

Notre modèle BiCharaConv, est entraîné avec les paramètres suivants :

– Une triplet loss avec 50 phrases (550 mots et environ $3M$ triplets) par batch avec une marge $\beta = 0.4$

– Un sous-échantillonnage des mots les plus fréquents

– Une longueur de mots de 25 ($l_w = 25$)

– Une représentation d'entrée avec les caractères représentés comme des vecteurs orthonormaux des 256 dimensions ($d_c = 256$).

– Un coefficient d'orthogonalisation γ de 0.1 (voir 6 pour plus d'informations)

Le modèle est entraîné pendant 200 epochs avec l'optimiseur Adam. Le pas d'apprentissage est de 10^{-4} durant tout le processus.

Il y a un total de 19 millions de paramètres à optimiser ce qui est faible comparé aux $2 \times N \times d$ paramètres de GloVe[2] avec N la taille du dictionaire et d la dimension de l'espace de représentation (soit 240 millions de paramètres à optimiser pour le modèle pris pour comparaison). Quant à fastText, il comporte davantage de paramètres que GloVe avec $(N + 2 \times 10^6) \times d$ soit 900 millions de paramètres pour le modèle utilisé pour les comparaisons, en ne comptant que l'encodeur. Notons toutefois que ces deux méthodes sont linéaires là où la notre est non-linéaire.

Le temps nécessaire pour l'entraînement est de 10 jours sur une carte GeForce GTX 1080 Ti de 11Go, une epoch durant en moyenne 1 heure et 12 minutes. En moyenne nous traitons un peu plus de 2500 mots par seconde avec cette configuration.

2. Deux vecteurs pour chaque mot, un pour quand le mot est le mot courant et un pour quand il est dans le contexte d'un autre mot. Les biais ne sont pas comptés dans le calcul.

5. Résultats

Le tableau 1 montre que notre méthode est aussi bonne que fastText mais avec des résultats plus homogènes sur tous les tests. En effet, les deux points forts de fastText sont la capacité à capturer le caractère concret dans l'espace de représentation et la capacité à prédire les caractéristiques alors que ses scores en matière de similarité/relation sont plus faibles. GloVe obtient de bons résultats en matière de similarité/relation, mais des scores plus faibles dans les deux autres méthodes d'évaluation. Notre méthode prend en compte à la fois le caractère concret, la similarité et les caractéristiques des mots dans l'espace de représentation.

Par ailleurs nos scores en concrétude sont quasiment au niveau de fastText ce qui prouve qu'une architecture convolutive sur les caractères peut être compétitive. Les produits bilinéaires participent aussi à ce résultat (voir section 6) même s'il y a de nombreux mots abstraits/concrets qui ne partagent pas nécessairement les mêmes racines étymologiques et donc pas les mêmes motifs de caractères. Cela tend à montrer que comme en vision par ordinateur, les produits bilinéaires, qui apportent une information de second ordre, permettent d'obtenir une amélioration des scores.

Test	GloVe	fastText	BiCharaConv
Concrétude	0.68	**0.71**	0.70
Similarité	**0.53**	0.45	0.52
Prédiction de caractéristiques	0.44	**0.52**	0.47
Moyenne	0.55	**0.56**	**0.56**

Tableau 1 – Comparaison entre GloVe, fastText et BiCharaConv. Le score de similarité est une moyenne des scores obtenus sur tous les jeux de données et le score de prédiction de caractéristiques et une moyenne sur toutes les connaissances

Les résultats de similarité de tous les jeux de données figurent dans le tableau 2. Nous voyons que si fastText capture la similarité (score élevé dans SimLex) et fait une bonne modélisation des mots rares (scores élevés dans le jeux de données RW, dont les paires contiennent au moins un mot rare), il ne capture pas la relation sémantique entre les mots. GloVe et notre méthode capturent mieux la relation sémantique mais nous ne faisons pas une représentation des mots rares aussi bonne que fastText. Il semble que les n-grammes permettent une meilleure représentation des mots rares que notre méthode basée sur des produits bilinéaires

Les résultats détaillés des prédictions de caractéristiques dans le tableau 3 montrent que notre modèle est le plus performant en ce qui concerne les prédictions relatives à la fonction, au son et aux caractéristiques tactiles. Cela est probablement dû au mélange choisi pour composer notre corpus. Fondamentalement, les phrases MS-COCO apportent probablement des informations qui nous aident à encoder ces notions.

Jeux de données	GloVe	fastText	BiCharaConv
MEN	**0.75**	0.60	0.72
MTURK771	0.65	0.44	**0.66**
RW	0.41	**0.53**	0.42
SemSim	**0.66**	0.62	0.63
SimLex	**0.37**	**0.37**	0.35
USF	**0.24**	0.12	0.21
VisSim	**0.53**	0.51	0.48
WS353	0.61	0.37	**0.67**

Tableau 2 – Scores détaillés de similarité/relation entre deux mots pour GloVe, fastText et notre modèle BiCharaConv

Knowledge	GloVe	fastText	BiCharaConv
encyclopaedic	0.50	**0.69**	0.53
function	0.52	0.55	**0.64**
sound	0.26	0.35	**0.40**
tactile	0.16	0.21	**0.29**
taste	0.43	**0.57**	0.52
taxonomic	0.79	**0.83**	0.65
visual-colour	0.28	**0.35**	0.29
visual-form and surface	0.50	**0.59**	0.46
visual-motion	0.48	**0.58**	0.41

Tableau 3 – Scores détaillés de prédiction de caractéristiques pour GloVe, fastText et BiCharaConv

6. Étude d'ablation

6.1. *Influence du produit bilinéaire*

Pour démontrer les avantages du produit bilinéaire (blocs BP), nous avons entraîné deux modèles : un correspondant à l'architecture plus petite, mais similaire à BiCharaConv, représentée sur la figure 3 et un correspondant à la même architecture mais avec les blocs BP remplacés par des convolutions 1D. En effet, l'architecture plus petite comprend 256 filtres pour chaque convolution au lieu de 512 et la dernière projection linéaire est remplacée par un *GlobalAveragePooling*. Par ailleurs, les Leaky ReLU sont remplacées par des ReLU. Nous avons choisi la même configuration d'entraînement que précédemment sans le sous-échantillonnage des mots les plus fréquents. Les résultats sont présentés dans le tableau 4. Il apparaît clairement que les produits bilinéaires donnent un gain sur l'ensemble des résultats - au moins 4% de gain et jusqu'à 9%. L'effet est plus visible sur les tâches de similarité et de prédictions de caractéristiques.

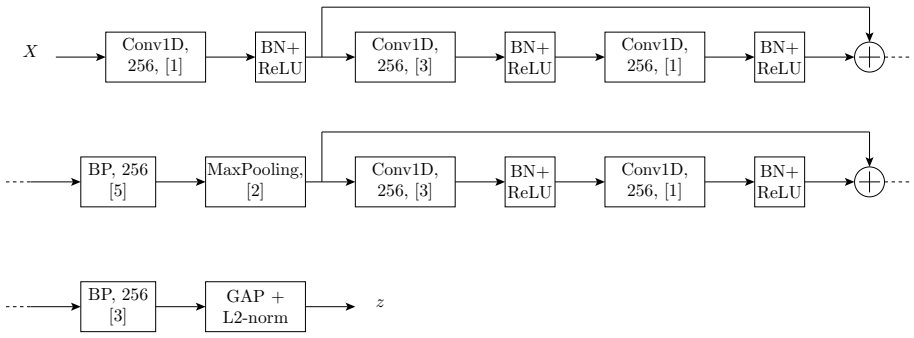

Figure 3 – Architecture utilisée par l'ablation sur le produit bilinéaire

	Concrétude	Similarité	Caractéristiques	Moyenne
Avec BP	**0.66**	**0.42**	**0.42**	**0.50**
Sans BP	0.62	0.35	0.33	0.43

Tableau 4 – Influence des produits bilinéaires

6.2. *Orthogonalisation*

Nous étudions ici l'impact de l'orthogonalisation. Nous considérons le modèle entier BiCharaConv représenté sur la figure 1. Notons qu'il n'y a pas sous-échantillonnage des mots les plus fréquents dans cette ablation.

Plus γ est élevé, plus l'information est équitablement répartie sur toutes les dimensions comme l'atteste la figure 4. Sans orthogonalisation ($\gamma = 0$), l'information n'est contenue que dans 50 dimensions ce qui prouve l'importance de cette dernière. Une valeur entre 1 et 10 nous permet d'avoir une occupation de l'espace similaire à celle de fastText.

Le tableau 5 montre les scores d'évaluation pour différentes valeurs de γ. Il semble que l'orthogonalisation ait un effet majeur sur l'évaluation de similarité. En outre, il semble que des valeurs de γ trop élevées détruisent l'espace de représentation même si les caractéristiques sont orthogonales. La combinaison du tableau 5 et de la figure 4 nous permet de choisir $\gamma = 0,1$. Cela nous donne le meilleur compromis entre l'orthogonalisation et l'intégration des scores d'évaluation.

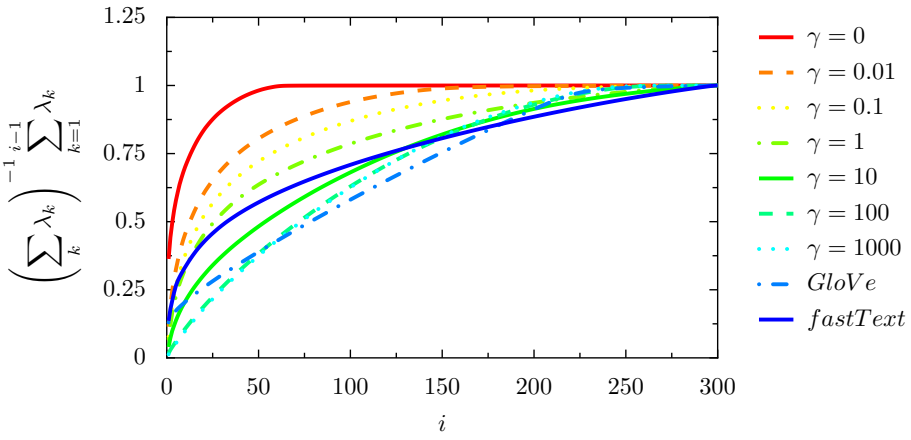

Figure 4 – Somme cumulée des valeurs propres de la matrice de covariance des mots d'évaluation dans l'espace de représentation pour : BiCharaConv avec différentes valeurs pour γ, GloVe et fastText. Meilleure visualisation en couleurs

γ	Concrétude	Similarité	Caractéristiques	Moyenne
0	0.66	0.47	0.42	0.52
0.01	**0.68**	0.50	0.43	**0.54**
0.1	0.67	0.50	**0.44**	**0.54**
1	0.66	**0.51**	0.43	0.53
10	0.63	0.50	0.39	0.51
100	0.36	0.26	0.19	0.27
1000	0.00	0.06	0.11	0.06

Tableau 5 – Scores pour plusieurs valeurs de γ

7. Conclusion

Dans ce travail, nous avons proposé une nouvelle approche pour apprendre des représentations de mots avec une entrée basée sur des caractères. Nous utilisons dans notre modèle BiCharaConv des convolutions directement sur les caractères, des connexions résiduelles pour faciliter le processus d'apprentissage et retenir les informations de bas niveau ainsi que des produits bilinéaires pour détecter les motifs de caractères. Notre méthode solutionne les problèmes inhérents aux méthodes à dictionnaire mais aussi à n-grammes et à convolutions puisqu'elle ne nécessite pas de décodeur. En premier lieu, nous résolvons le problème des mots hors vocabulaire et des fautes de frappe, deuxièmement, en traduisant l' *hypothèse de distribution* en un objectif d'apprentissage de métrique, nous évitons l'apprentissage d'un décodeur qui est inutilisé après l'entraînement tout en restant aussi efficace grâce aux produits bi-

linéaires. À l'avenir, il nous faudra explorer si les produits bilinéaires peuvent être intégrés dans des architectures de type *transformers* pour voir si, comme pour l'encodage de mots, les produits bilinéaires peuvent améliorer l'encodage de phrases.

8. Remerciements

Ce travail est réalisé dans le cadre d'une thèse co-financée par l'Agence Innovation Défense (AID) et l'École des Ponts ParisTech.

9. Bibliographie

Bahdanau D., Cho K., Bengio Y., « Neural Machine Translation by Jointly Learning to Align and Translate », *ICLR 2015*, 2015.

Bojanowski P., Grave E., Joulin A., Mikolov T., « Enriching Word Vectors with Subword Information », *Transactions of the Association for Computational Linguistics*, vol. 5, p. 135-146, 2017.

Bordes A., Chopra S., Weston J., « Question Answering with Subgraph Embeddings », *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, p. 615-620, 2014.

Bruni E., Tran N.-K., Baroni M., « Multimodal distributional semantics », *Journal of Artificial Intelligence Research*, vol. 49, p. 1-47, 2014.

Devlin J., Chang M.-W., Lee K., Toutanova K., « BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding », *arXiv preprint arXiv :1810.04805*, 2018.

Finkelstein L., Gabrilovich E., Matias Y., Rivlin E., Solan Z., Wolfman G., Ruppin E., « Placing search in context : The concept revisited », *ACM Transactions on information systems*, vol. 20, n° 1, p. 116-131, 2002.

Frome A., Singer Y., Sha F., Malik J., « Learning globally-consistent local distance functions for shape-based image retrieval and classification », *2007 IEEE 11th International Conference on Computer Vision*, IEEE, p. 1-8, 2007.

Gao Y., Beijbom O., Zhang N., Darrell T., « Compact bilinear pooling », *Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 317-326, 2016.

Halawi G., Dror G., Gabrilovich E., Koren Y., « Large-scale learning of word relatedness with constraints », *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, p. 1406-1414, 2012.

Harris Z. S., « Distributional structure », *Word*, vol. 10, n° 2-3, p. 146-162, 1954.

Hill F., Reichart R., Korhonen A., « Simlex-999 : Evaluating semantic models with (genuine) similarity estimation », *Computational Linguistics*, vol. 41, n° 4, p. 665-695, 2015.

Kim J., On K. W., Lim W., Kim J., Ha J., Zhang B., « Hadamard Product for Low-rank Bilinear Pooling », *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017.

Kim Y., Jernite Y., Sontag D., Rush A. M., « Character-aware neural language models », *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Kong S., Fowlkes C., « Low-rank bilinear pooling for fine-grained classification », *Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 365-374, 2017.

Lin T.-Y., Maire M., Belongie S., Hays J., Perona P., Ramanan D., Dollár P., Zitnick C. L., « Microsoft coco : Common objects in context », *European conference on computer vision*, Springer, p. 740-755, 2014.

Lin T.-Y., RoyChowdhury A., Maji S., « Bilinear cnn models for fine-grained visual recognition », *Proceedings of the IEEE international conference on computer vision*, p. 1449-1457, 2015.

Luong T., Socher R., Manning C., « Better word representations with recursive neural networks for morphology », *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, p. 104-113, 2013.

Lushan Han, Abhay L. Kashyap, Tim Finin, James Mayfield, and Johnathan Weese, « UMBC_EBIQUITY-CORE : Semantic Textual Similarity Systems », *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*, Association for Computational Linguistics, June, 2013.

Maas A., Daly R. E., Pham P. T., Huang D., Ng A. Y., Potts C., « Learning word vectors for sentiment analysis », *Proceedings of the 49th annual meeting of the association for computational linguistics : Human language technologies*, p. 142-150, 2011.

McCann B., Bradbury J., Xiong C., Socher R., « Learned in translation : Contextualized word vectors », *Advances in Neural Information Processing Systems*, p. 6294-6305, 2017.

McRae K., Cree G. S., Seidenberg M. S., McNorgan C., « Semantic feature production norms for a large set of living and nonliving things », *Behavior research methods*, vol. 37, nº 4, p. 547-559, 2005.

Mikolov T., Chen K., Corrado G., Dean J., « Efficient estimation of word representations in vector space », *ICLR Workshop*, 2013a.

Mikolov T., Grave E., Bojanowski P., Puhrsch C., Joulin A., « Advances in Pre-Training Distributed Word Representations », *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, 2018.

Mikolov T., Sutskever I., Chen K., Corrado G. S., Dean J., « Distributed Representations of Words and Phrases and their Compositionality », *in* C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, K. Q. Weinberger (eds), *Advances in Neural Information Processing Systems 26*, Curran Associates, Inc., p. 3111-3119, 2013b.

Nelson D. L., McEvoy C. L., Schreiber T. A., « The University of South Florida free association, rhyme, and word fragment norms », *Behavior Research Methods, Instruments, & Computers*, vol. 36, nº 3, p. 402-407, 2004.

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

Peters M. E., Neumann M., Iyyer M., Gardner M., Clark C., Lee K., Zettlemoyer L., « Deep contextualized word representations », *NAACL*, 2018.

Sennrich R., Haddow B., Birch A., « Neural Machine Translation of Rare Words with Subword Units », *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, Association for Computational Linguistics, Berlin, Germany, p. 1715-1725, August, 2016.

Silberer C., Lapata M., « Learning grounded meaning representations with autoencoders », *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, vol. 1, p. 721-732, 2014.

Srivastava R. K., Greff K., Schmidhuber J., « Training very deep networks », *Advances in neural information processing systems*, p. 2377-2385, 2015.

Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser Ł., Polosukhin I., « Attention is all you need », *Proceedings of the 31st International Conference on Neural Information Processing Systems*, p. 6000-6010, 2017.

Wang A., Singh A., Michael J., Hill F., Levy O., Bowman S., « GLUE : A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding », *Proceedings of the 2018 EMNLP Workshop BlackboxNLP : Analyzing and Interpreting Neural Networks for NLP*, p. 353-355, 2018.

Wang C., Cho K., Gu J., « Neural machine translation with byte-level subwords », *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, p. 9154-9160, 2020.

Wieting J., Bansal M., Gimpel K., Livescu K., « Charagram : Embedding Words and Sentences via Character n-grams », *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, p. 1504-1515, 2016.

Wu Y., Schuster M., Chen Z., Le Q. V., Norouzi M., Macherey W., Krikun M., Cao Y., Gao Q., Macherey K. *et al.*, « Google's neural machine translation system : Bridging the gap between human and machine translation », *arXiv preprint arXiv :1609.08144*, 2016.

Zablocki E., Piwowarski B., Soulier L., Gallinari P., « Learning multi-modal word representation grounded in visual context », *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

Zhang X., Zhao J., LeCun Y., « Character-level Convolutional Networks for Text Classification », *in* C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, R. Garnett (eds), *Advances in Neural Information Processing Systems 28*, Curran Associates, Inc., p. 649-657, 2015.