Réseaux convolutionnels et modèles de caractères

Matthieu Labeau

2017-12-12

Outline

Plan

Modèles de langue récurrent: Rappel

But

Estimer les probabilités **non-nulles** d'une séquence de mots étant donné un vocabulaire $\mathcal{V}$:

$$P(w_1^L) = P(w_1, w_2, ...w_L) = \prod_{i=1}^{L} P(w_i|w_1^{i-1}), \quad \forall i, w_i \in \mathcal{V}$$

de manière **récurrente**:

$$P(w_i|w_1^{i-1})$$

Caractéristiques de la structure récurrente:

- Calcule de gauche à droite: on a besoin d'un contexte préfixe
- L'information la plus récente est la mieux (trop ?) retenue
- Relations (et calculs de gradients) compliqués

Autres tâches

Modélisation de séquence

Par exemple, pour la classification:

- Bag of words, Continuous BOW: Efficace, mais pas de gestion des combinaisons, de l'ordre des mots
- Bag of n-grams: Résout le problème, mais pas de partage des représentations pour mots proches, et explosion du nombre de paramètres
- Réseaux réccurents ?

Tagging de séquence

Par exemple, Part of Speech (POS) Tagging:

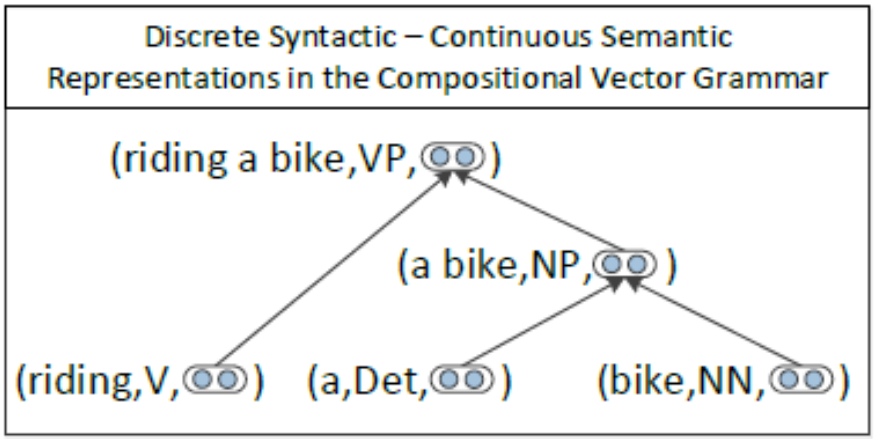
- Bag-of words/Ngrams: même problèmes que précédemment
- Feedforward NN, Reccurent NN: même problèmes

mais aussi Named Entity recognition (NER), Semantic Role Labelling, ...

Comment extraire efficacement des caractéristiques (features) utiles sur l'ensemble de la phrase ?

Plan

Aparté: Réseaux récursifs



- Naturellement adaptée aux phrases
- Besoin d'obtenir la structure d'arbre syntactique associée

Références: (Socher et al.2011; Richard Socher and John Bauer and Christopher D. Manning and Andrew Y. Ng2013; Socher et al.2014)

Réseaux convolutionnels

Qu'est ce qu'une convolution ?

- Application d'un **filtre** sur un **signal d'entrée**
- Le signal et le filtre peuvent être des **vecteurs** ou des **matrices** (images)
- On applique le filtre sur chacune des fenêtres du signal de taille correspondante.

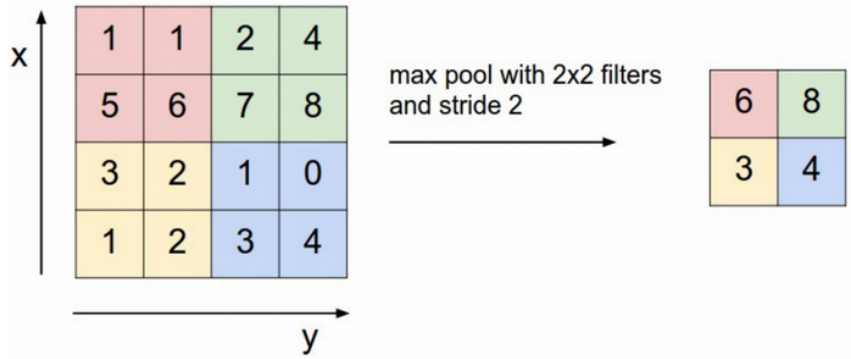
CNN: Convolutional neural networks

- On utilise des convolutions sur l'entrée pour calculer la sortie
- Chaque couche applique une grande quantité de filtres, chaque application résultant en une valeur
- Les résultats pour chaque partie de l'entrée sont combinés avec du *pooling*
- Les filtres sont les paramètres du réseau, appris automatiquement
- Avantage: ils sont très **rapides**

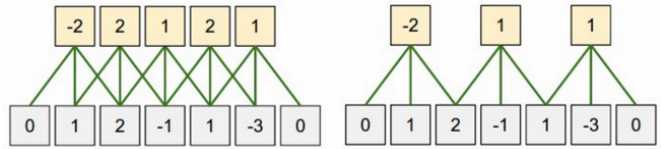
Réseaux convolutionnels: fonctionnement

Choix:

- **Pooling**:

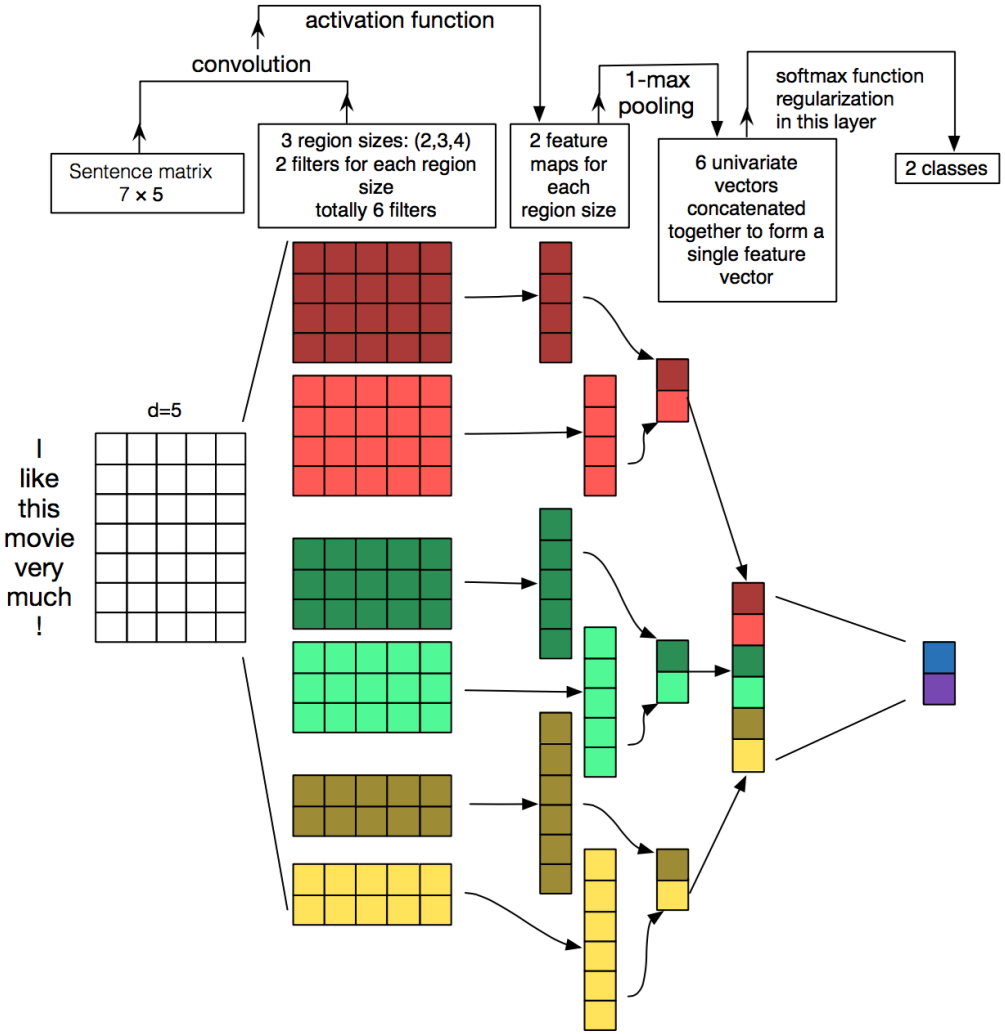


- **Stride**:

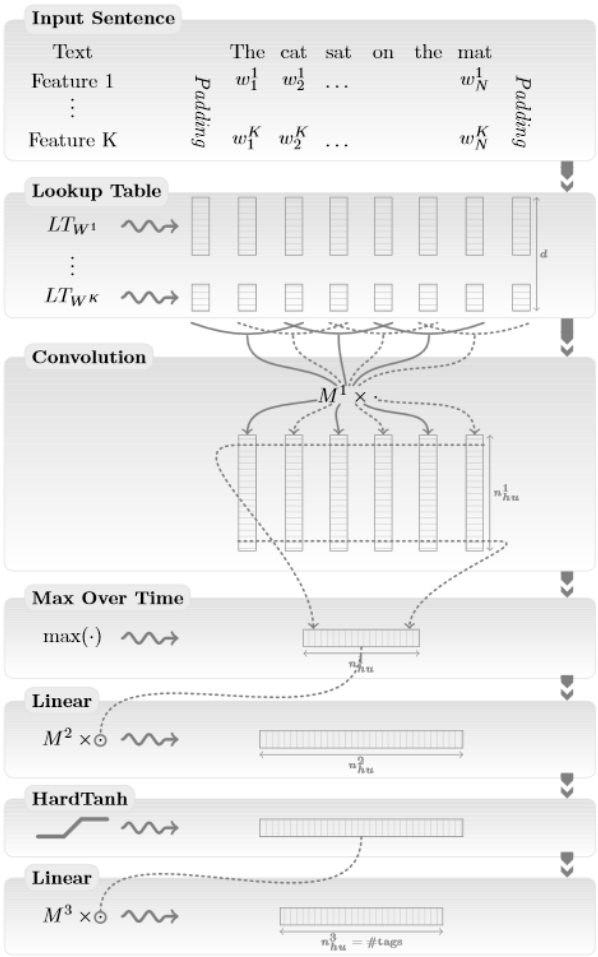


- **Padding**: Gestion des bords

Réseaux convolutionnels: Exemple

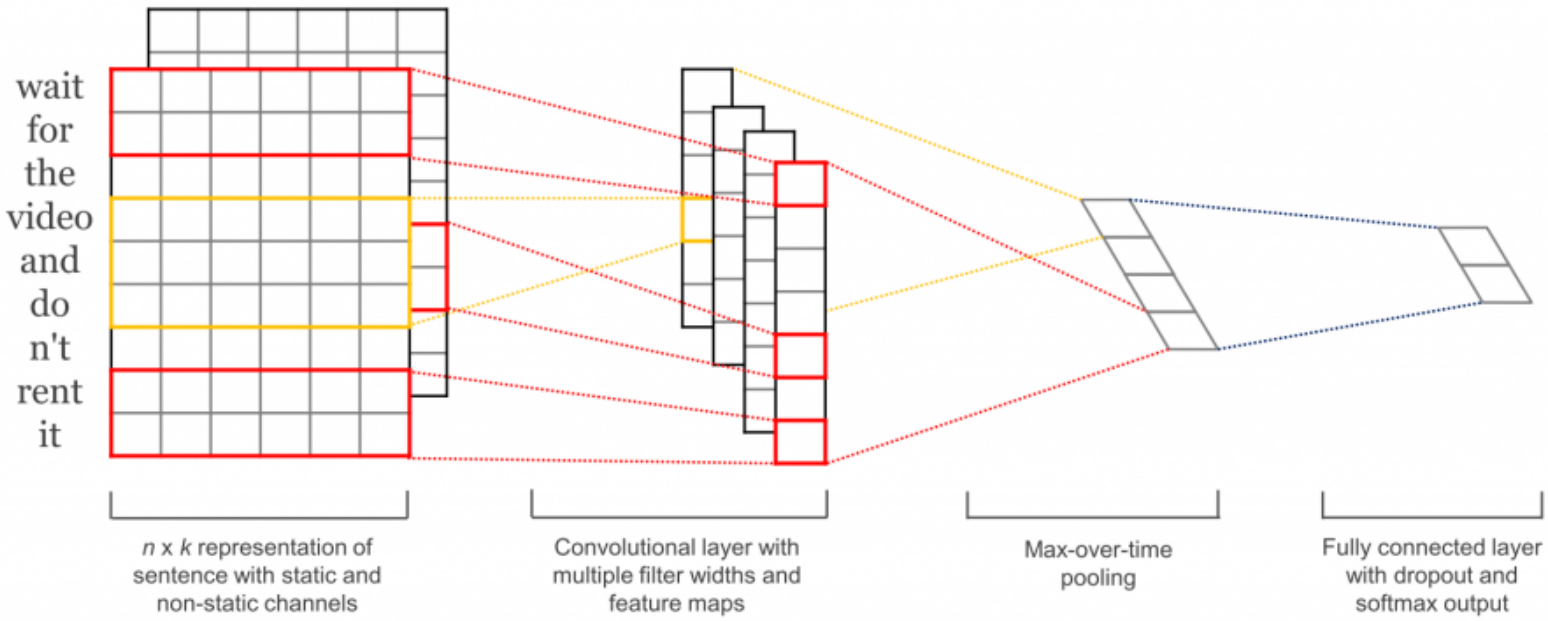


Premières applications: (Collobert et al.2011)



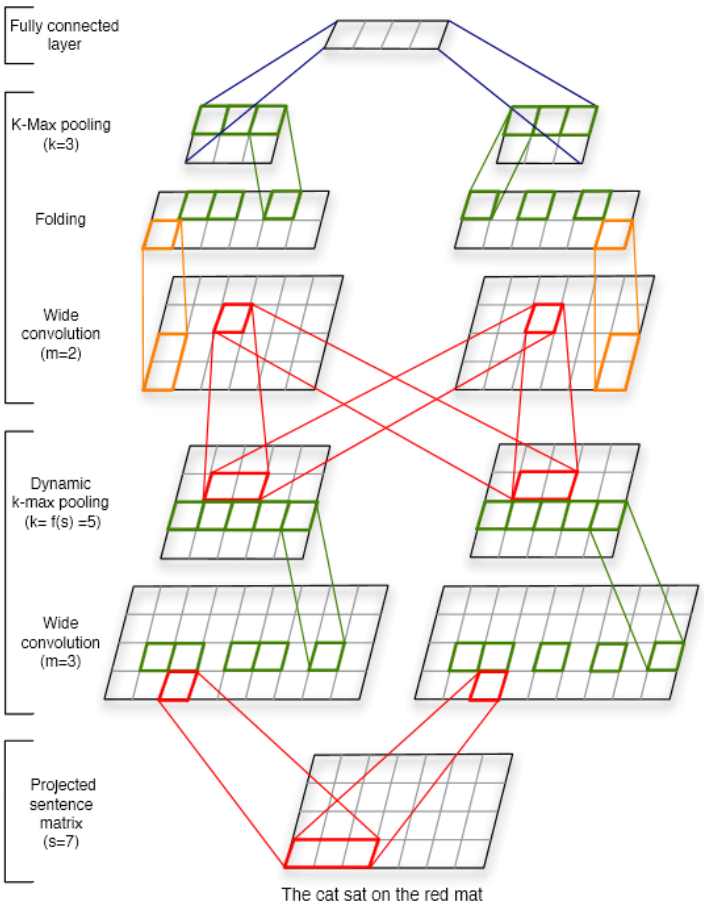
- Architecture unifiée pour tâches de tagging sur séquence
- Permet une extraction automatique de features appropriées sur l'ensemble de la phrase

Premières applications: (Kim2014)



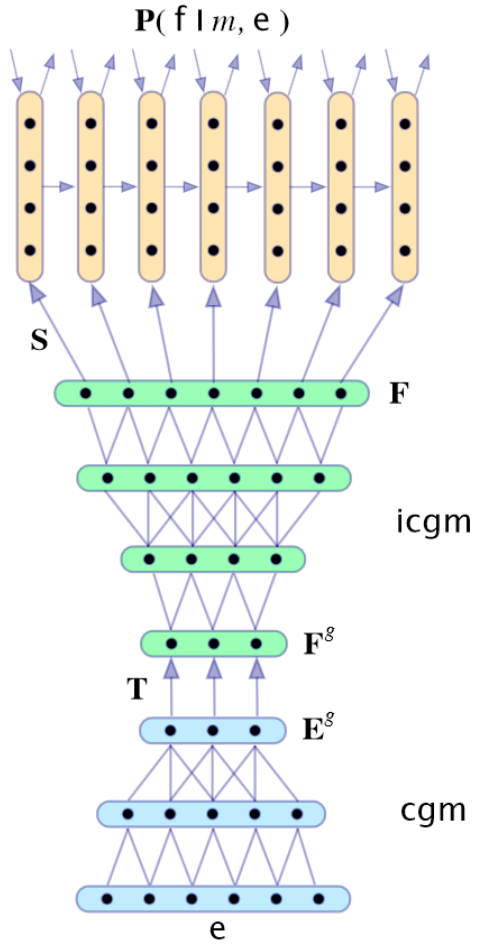
- Permet d'obtenir simplement une représentation de la phrase sensible à l'odre des mots
- Appliqué a plusieurs tâches: classification binaire, prédiction de sentiment, classification multinomiale de questions

Modélisation de séquences: (Kalchbrenner et al.2014)



- Structure un peu plus complexe (avec *folding*)
- Permet d'inférer un graphe qui "remplace" l'arbre de parse

Premiers modèles de traduction neuronale:(Kalchbrenner and Blunsom2013)



- Premier modèle fonctionnel de traduction neuronale
- Modèle convolutionel pour encoder les phrases, et récurrent pour les décoder

Plan

Modèles de langue: Rappel

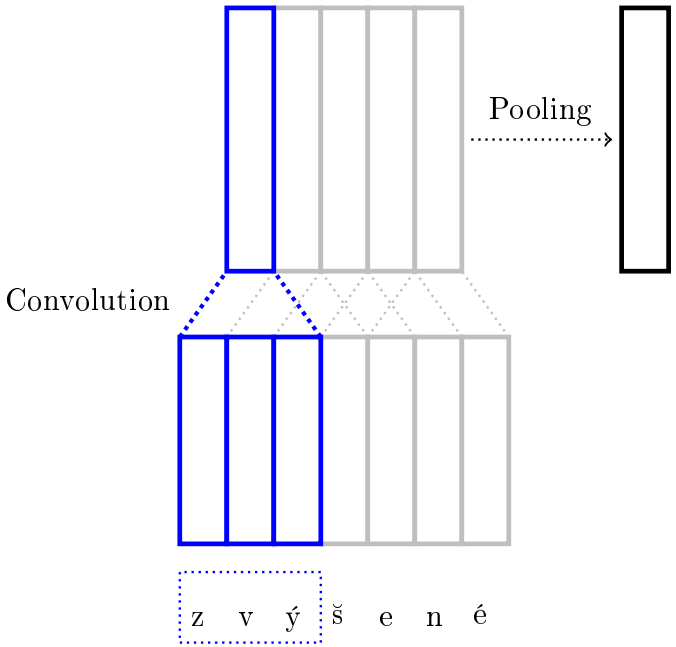
Représentations:

- La plupart des modèles de langue neuronaux, tels que les modèles n-grammes, modélisent les mots par des vecteurs de valeurs réelles
- Un vocabulaire fini $\mathcal{V}$ contient la liste des mots dont le modèle paramétrise la représentation - ils sont regroupés dans une table de paramètres (*look-up table*) $\mathbf{L}$ de dimension $|\mathcal{V}| \times d_r$, en entrée et en sortie

Inconvénients:

- Pour les langues morphologiquement riches (Tchèque, Allemand...) couvrir l'ensemble du lexique se révèle difficile, à cause de l'explosion combinatoire du nombre de mots
- La structure des mots est ignorée: le modèle attribue des paramètres à la modélisation de ce qui pourrait plus efficacement être appréhendé en décomposant le mot

Représentation des caractères

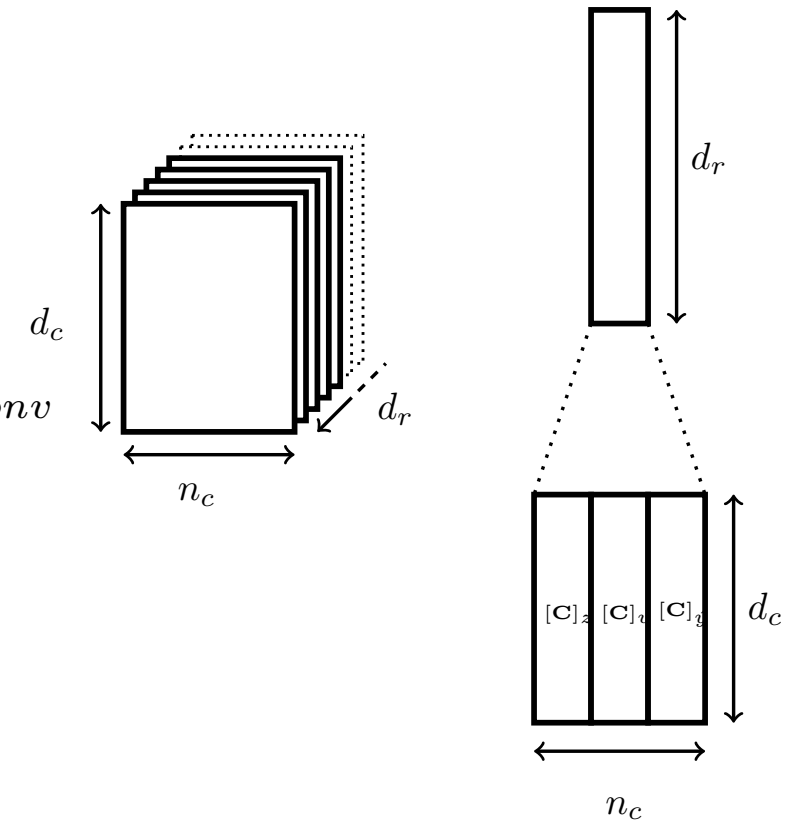


Les représentations vectorielles des caractères sont stockées dans une table de paramètres $\mathbf{C}$
On utilise une convolution sur les n-grams de caracatères, puis du *pooling* sur les vecteurs obtenus, pour obtenir la représentation du mot

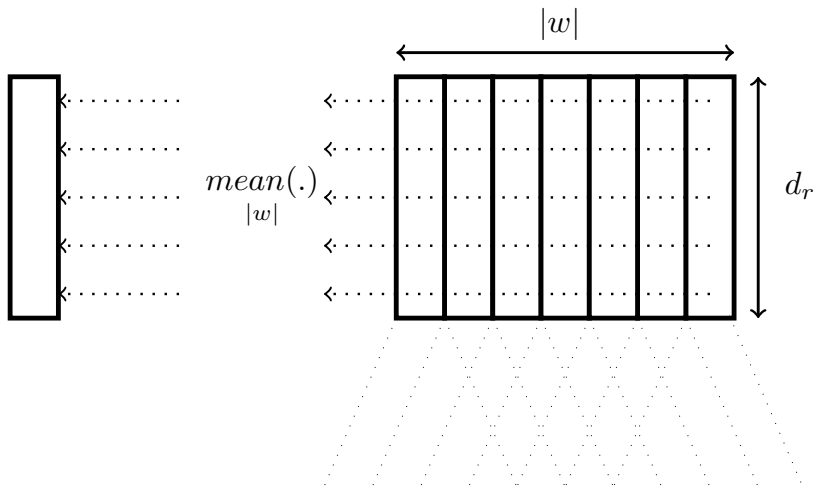
Couche de convolution

On applique d_r filtres de convolution $[\mathbf{W}_k^{conv}]_{k=1..d_r}$, chacun étant de dimension $d_c \times n_c$ à une fenêtre glissante de n_c caractères, chacun produisant une caractéristique locale:

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



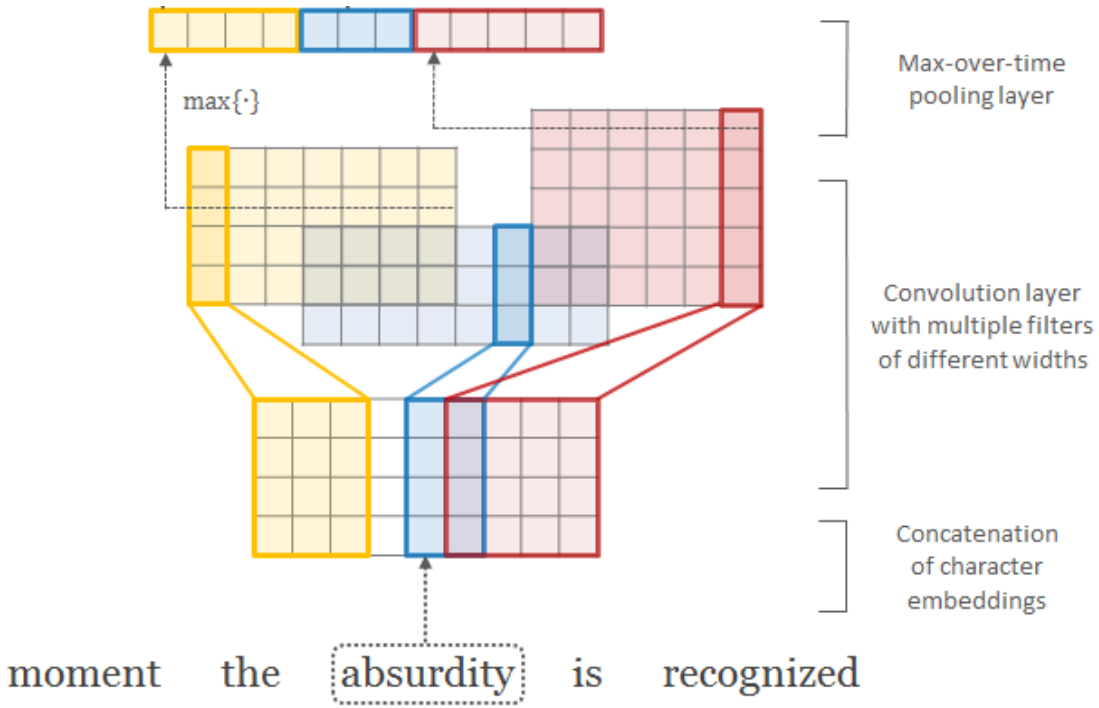
Couche de convolution: *pooling*



On calcule la moyenne des i-èmes éléments des vecteurs obtenus, à laquelle on applique la fonction d'activation ϕ :

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

Character-Aware Neural Language Models (Kim et al.2015)



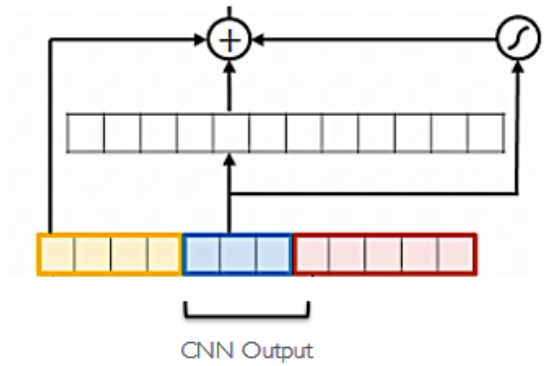
- Convolution sur des embeddings de caractères avec des filtres de différentes tailles
- Le *Max-pooling* revient à selectionner les n-grams (de différentes tailles)

Aparté: Highway Networks (Srivastava et al.2015)

$$\mathbf{t} = \sigma(\mathbf{W}_T\mathbf{y} + \mathbf{b}_T)$$

$$\mathbf{z} = \mathbf{t} \odot g(\mathbf{W}_H\mathbf{y} + \mathbf{b}_H) + (\mathbf{1} - \mathbf{t}) \odot \mathbf{y}$$

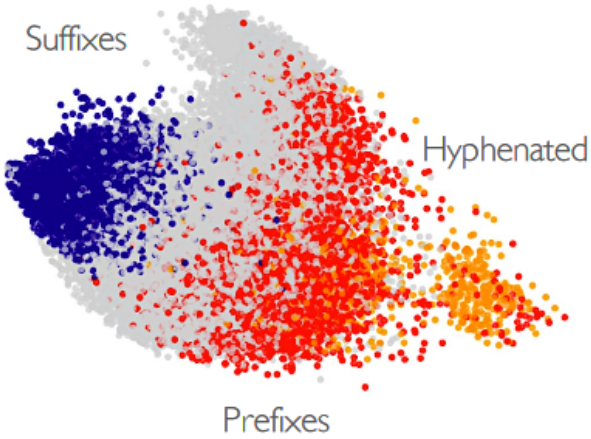
Transform Gate — Input — Carry Gate



- Permet de transformer une partie des informations alors qu'une autre partie est directement transmise
- Utilisé pour "mélanger" différentes représentations (ici, taille de n-grams différentes)

Character-Aware Neural Language Models (Kim et al.2015)

Out-of-Vocabulary		
computer-aided	*misinformed*	*looooook*
–	–	–
–	–	–
–	–	–
–	–	–
computer-guided	*informed*	*look*
computerized	*performed*	*cook*
disk-drive	*transformed*	*looks*
computer	*inform*	*shook*
computer-guided	*informed*	*look*
computer-driven	*performed*	*looks*
computerized	*outperformed*	*looked*
computer	*transformed*	*looking*



- On utilise un LSTM + Hierarchical softmax
- Résultats comparable à l'état de l'art avec beaucoup moins de paramètres

Références et ressources

- http://web.stanford.edu/class/cs224n/lectures/cs224n-2017-lecture13-CNNs.pdf
- http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/
- https://phontron.com/class/nn4nlp2017/index.html


Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa.

2011.

Natural language processing (almost) from scratch.

CoRR, abs/1103.0398.


Nal Kalchbrenner and Phil Blunsom.

2013.

Recurrent continuous translation models.

In *EMNLP*, pages 1700–1709. ACL.


Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom.

2014.

A convolutional neural network for modelling sentences.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, June.


Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush.

2015.

Character-aware neural language models.

CoRR, abs/1508.06615.


Yoon Kim.

2014.

Convolutional neural networks for sentence classification.

In *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*, pages 1746–1751.

Richard Socher and John Bauer and Christopher D. Manning and Andrew Y. Ng.

2013.

Parsing With Compositional Vector Grammars.

In *ACL*.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning.

2011.

Dynamic pooling and unfolding recursive autoencoders for paraphrase detection.

Richard Socher, Andrej Karpathy, Quoc V. Le, Christopher D. Manning, and Andrew Y. Ng.

2014.

Grounded compositional semantics for finding and describing images with sentences.

TACL, 2:207–218.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber.

2015.

Highway networks.

CoRR, abs/1505.00387.