
Modèles de langue

Introduction

Estimer la probabilité du prochain mot dans une phrase

Jean se dirige vers le bar pour commander un verre de ...

Introduction

Traditionnellement

1. On créer une matrice de probabilité
 - a. Probabilité conditionnelle $P(\textit{whisky} \mid \textit{un verre de})$
 - b. On utilise les n-grams pour construire cette matrice

$$P(\textit{whisky} \mid \textit{un verre de}) = \#(\textit{un, verre, de, whisky}) / \#(\textit{un, verre, de})$$

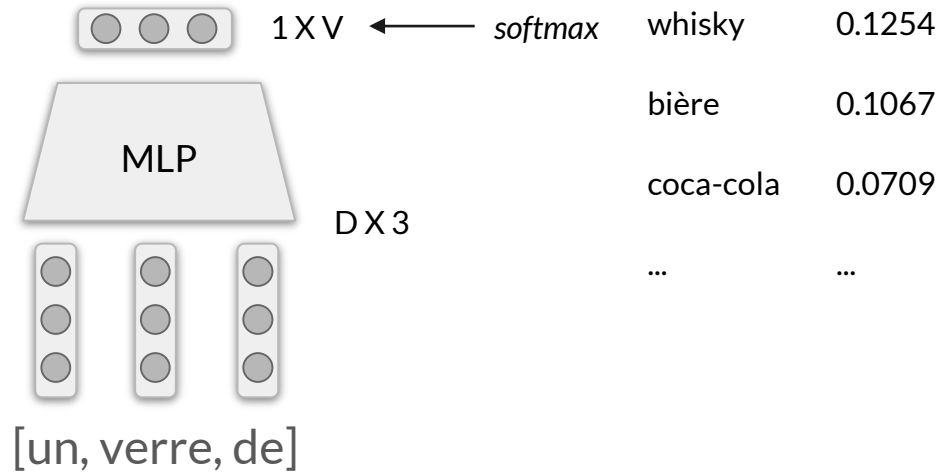
Évaluation

1. On utilise la cross-entropy
2. Pour calculer la perplexité de notre modèle:
 - a. $\Rightarrow 2^{\text{cross-entropy}}$
 - b. Défini l'incertitude du modèle
 - c. Une perplexité élevée \Rightarrow Modèle incertain

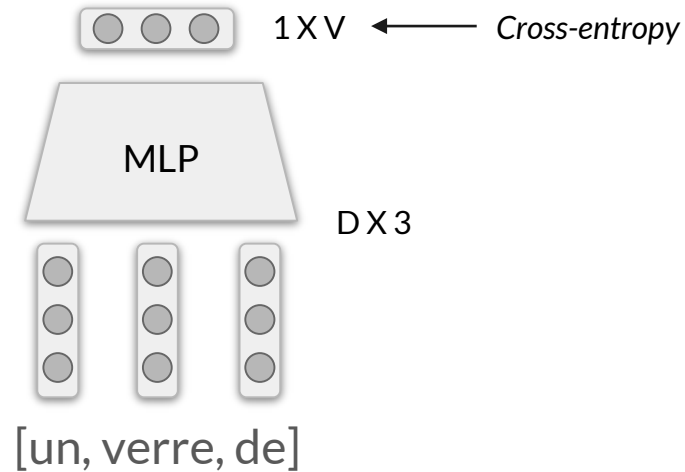
Modèle de langue neuronal

1. Réutilisation du concept de n -grams
 - a. Fenêtre fixe de n mots pour prédire le $n+1$
2. Gère naturellement les n -grams rarement/jamais vu en train/test

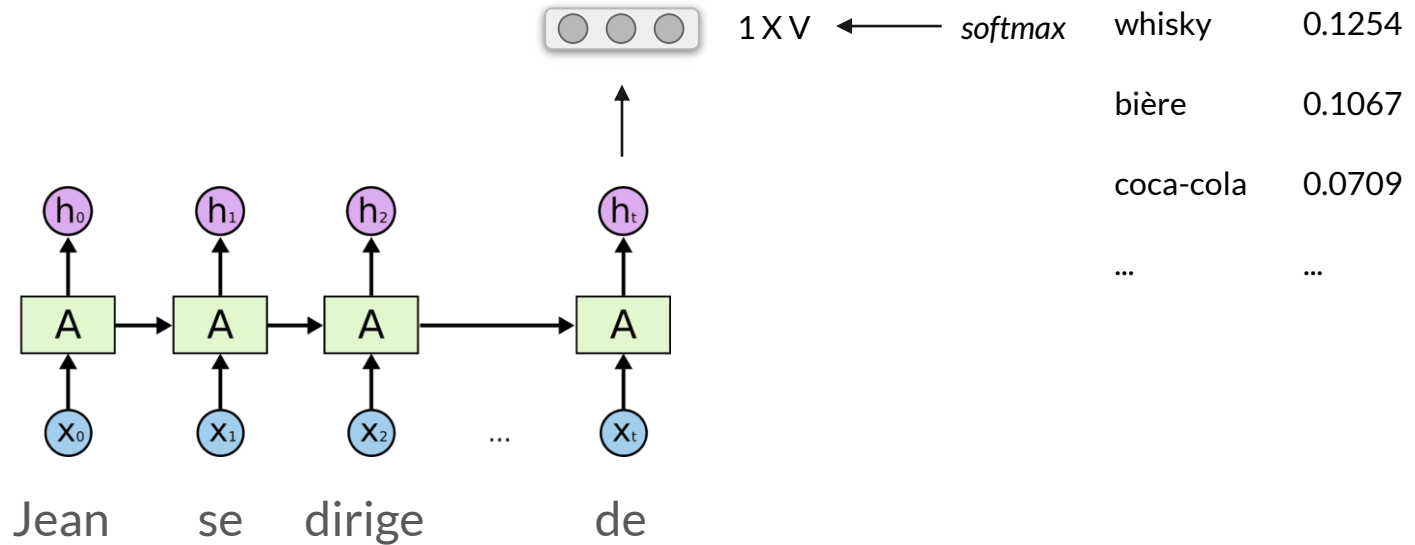
Modèle de langue neuronal



Modèle de langue neuronal



Modèle de langue RNN



Modèle de langue RNN

1. On peut appliquer le même modèle sur les caractères
2. Softmax sur l'ensemble des caractères possible
 - a. Plus léger computationnellement
3. Génération de mots jamais vu dans le corpus

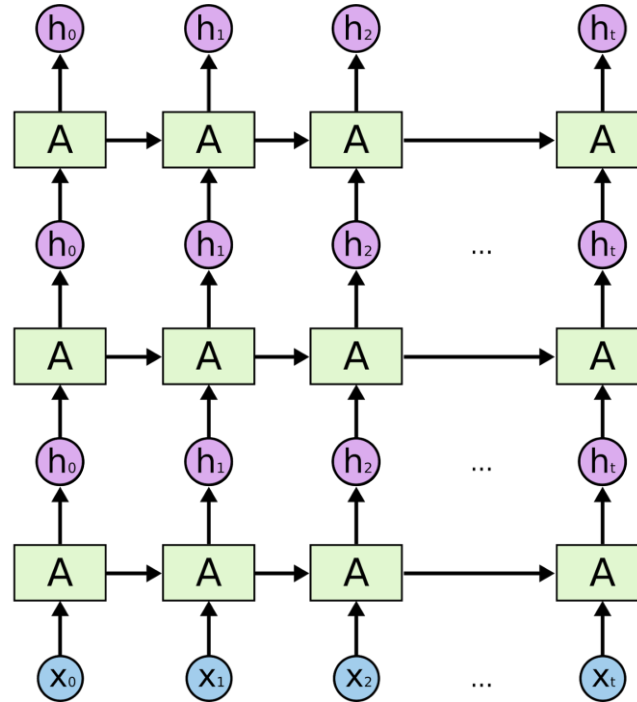
Deep RNNs

Appliqué aux modèles de langue

Deep RNNs

1. On a vu des réseaux CNN très profond
 - a. Resnet 152 couches
2. RNN (et LSTM) beaucoup plus dure à entraîner
 - a. On va parler plutôt de 2-3 couches

Deep RNNs



Génération de langue

*Lets train a 2-layer LSTM with 512 hidden nodes (approx. 3.5 million parameters), and with **dropout of 0.5 after each layer**. We'll train with **batches of 100 examples and truncated backpropagation through time of length 100 characters**.*

Génération de langue

“The surprised in investors weren’t going to raise money. I’m not the company with the time there are all interesting quickly, don’t have to get off the same programmers.”

Comment on génère le prochain mot?

1. On prend le mot avec la probabilité maximale
 - a. Génération déterministe, sans variété
2. On pige dans la distribution obtenue par la *softmax*
 - a. Génération plus variée

Génération conservatrice

“is that they were all the same thing that was a startup is that they were all the same thing that was a startup is that they were all the same thing that was a startup is that they were all the same”

Plus d'exemples

<http://karpathy.github.io/2015/05/21/rnn-effectiveness/>

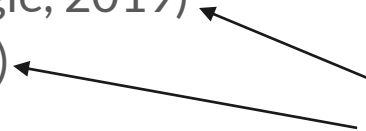
Enjeux en génération de langue

1. Au delà du maximum likelihood (Cross-Entropy) à l'entraînement
 - a. Reinforcement learning
 - b. Entraînement adversarial (GAN)
2. Génération à partir de templates
3. Teacher-Forcing, Scheduled Sampling, autre...?
4. Conditionnement à la génération
 - a. Générer selon un style, un thème, un contexte...
5. Évaluation des modèles

État de l'art dans la modélisation du langage

1. AWD-LSTM (SalesForce, 2017)
2. Transformer XL (Google, 2019)
3. GPT-2 (OpenAI, 2019)

Architecture transformer!



AWD-LSTM

1. Embeddings de mots en entrée (non pré-entraîné!)
2. 3 couches de LSTM uni-directionnels
3. Beaucoup de travail fait sur la régularisation

AWD-LSTM

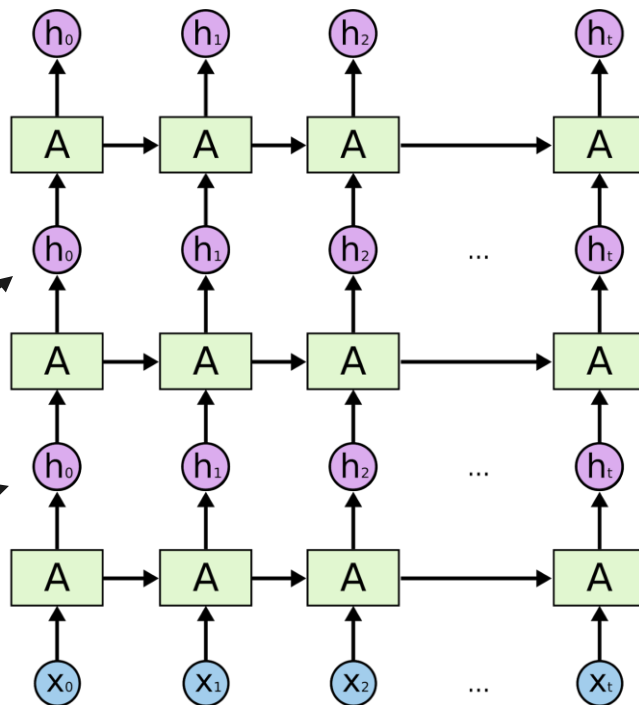
1. Régularisation:

- a. Dropout sur les embeddings
- b. BPTT de longueur variable
- c. Dropout sur les *hidden states*
- d. Weight Dropout
- e. Weight Tying
 - i. Entre la couche d'embeddings et les poids avant la softmax
 - ii. Réduit de beaucoup le nombre de paramètres

AWD-LSTM

Dropout sur les hidden states

Dropout appliqué à ces niveaux



AWD-LSTM

Weight Dropout appliqué sur les paramètres hidden-to-hidden

$$\begin{aligned}i_t &= \sigma(W^i x_t + U^i h_{t-1}) \\f_t &= \sigma(W^f x_t + U^f h_{t-1}) \\o_t &= \sigma(W^o x_t + U^o h_{t-1}) \\\tilde{c}_t &= \tanh(W^c x_t + U^c h_{t-1}) \\c_t &= i_t \odot \tilde{c}_t + f_t \odot \tilde{c}_{t-1} \\h_t &= o_t \odot \tanh(c_t)\end{aligned}$$

Pour prévenir l'overfitting
sur la **connexion récurrente**

AWD-LSTM

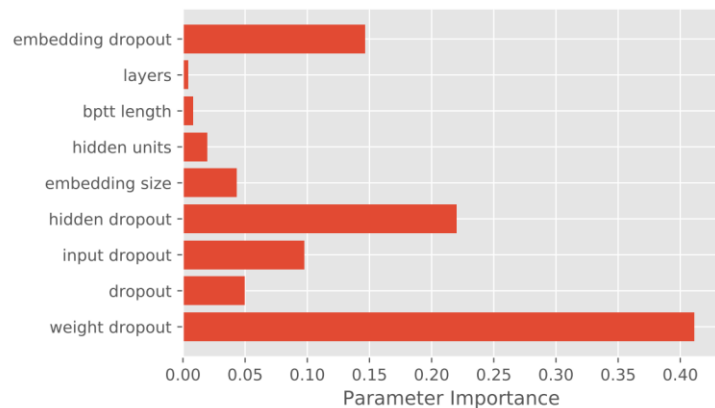
Le même masque est appliqué sur les paramètres hidden-to-hidden tout au long de la séquence

$$\begin{aligned}i_t &= \sigma(W^i x_t + U^i h_{t-1}) \\f_t &= \sigma(W^f x_t + U^f h_{t-1}) \\o_t &= \sigma(W^o x_t + U^o h_{t-1}) \\\tilde{c}_t &= \tanh(W^c x_t + U^c h_{t-1}) \\c_t &= i_t \odot \tilde{c}_t + f_t \odot \tilde{c}_{t-1} \\h_t &= o_t \odot \tanh(c_t)\end{aligned}$$

Pour prévenir l'overfitting sur la **connexion récurrente**

AWD-LSTM

Effet des différentes méthodes de régularisation



Autres modèles

1. Transformer-XL
2. GPT-2
3. BERT*
4. Nous les verrons lorsque nous aurons introduit les modèles d'attention...

Importance des modèles de langues

1. Utilisé pour de la génération textuelle
2. Aussi dans plusieurs autres modèles de NLP
 - a. Traduction automatique
 - b. Question answering
 - c. Reading Comprehension
 - d. Summarization
 - e. et plus...

ELMo

ELMo

1. Modèle de langue bidirectionnel
2. Remplace une couche de word embeddings
 - a. *So long word2vec!*
3. Génère des embeddings spécialisés pour des tâches précises



ELMo

On commence par une modélisation **par token** (mot):

1. Soit on prend une table d'embeddings
2. Soit on modélise un mot à l'aide d'un réseau de neurones à convolution sur les caractères
 - a. ELMo utilise un CNN
3. On obtient une représentation \mathbf{x}_k^{LM}

ELMo

Modèle de langue bidirectionnel

1. Modèle de langue “forward”:
 - a. Pour une séquence de N mots;
 - b. On calcul la probabilité de chacun des mots de la phrases étant donné son historique

$$p(t_1, t_2, \dots, t_N) = \prod_{k=1}^N p(t_k \mid t_1, t_2, \dots, t_{k-1}).$$

ELMo

Modèle de langue bidirectionnel

1. Modèle de langue “backward”:
 - a. Même chose que le modèle “forward” mais on envoie la séquence de mots à l’envers

$$p(t_1, t_2, \dots, t_N) = \prod_{k=1}^N p(t_k \mid t_{k+1}, t_{k+2}, \dots, t_N).$$

ELMo

Un modèle de langue bidirectionnel (*biLM*) va donc combiner les 2 représentations obtenues à l'aide des modèles “forward” et “backward”.

Pour un modèle à L couches *biLM*, ELMo va générer $2L+1$ représentations

$$R_k = \{ \mathbf{x}_k^{LM}, \overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} \mid j = 1, \dots, L \}$$

ELMo

L'utilisation des représentations se fait de 2 manières:

1. On utilise simplement les hiddens de la dernière couche
2. On apprend à combiner les différentes représentations

$$\mathbf{h}_{k,L}^{LM}$$

$$E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^L s_j^{task} \mathbf{h}_{k,j}^{LM} .$$

Paramètre pour scaler le
vecteur selon la tâche

Paramètre appris par couche de
représentation

ELMo

Workflow d'utilisation de ELMo dans un autre modèle:

1. On pré-entraîne le modèle de langue bidirectionnel sur beaucoup de données
2. On l'utilise dans les architectures existantes pour des tâches spécifiques;
 - a. On freeze les poids des couches biLM
 - b. On obtient la représentation ELMo pour chaque mot
 - c. On les concatène pour les envoyer dans le modèle spécifique à la tâche

ELMo

Workflow d'utilisation de ELMo dans un autre modèle:

$$[\mathbf{x}_k; \mathbf{ELMo}_k^{task}]$$

On peut également dégeler les couches de ELMo pour le fine-tuner pour la tâche!

To Tune or Not to Tune?

Adapting Pretrained Representations to Diverse Tasks

Matthew Peters^{1*}, Sebastian Ruder^{2,3†*}, and Noah A. Smith^{1,4}

¹Allen Institute for Artificial Intelligence, Seattle, USA

²Insight Research Centre, National University of Ireland, Galway, Ireland

³Aylien Ltd., Dublin, Ireland

⁴Paul G. Allen School of CSE, University of Washington, Seattle, USA

{matthewp, noah}@allenai.org, sebastian@ruder.io

ELMo

Évaluation / Intégration du modèle: 6 tâches;

1. Question Answering
2. Textual entailment
3. Semantic Role Labeling
4. Coreference Resolution
5. Named Entity Recognition
6. Sentiment Analysis

ELMo

<https://demo.allennlp.org>

ELMo

TASK	PREVIOUS SOTA		OUR BASELINE	ELMo + BASELINE	INCREASE (ABSOLUTE/ RELATIVE)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 \pm 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 \pm 0.19	90.15	92.22 \pm 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 \pm 0.5	3.3 / 6.8%

ELMo

Résultats intéressants:

1. Les représentations biLM à **la tête** du réseaux améliorent les tâches plutôt **sémantique** (Word Sense Disambiguation)
2. Les représentations biLM à **la base** du réseaux améliorent les tâches plutôt **syntaxique** (Part of Speech Tagging)

ELMo

Avantages intéressants:

1. S'incorpore à n'importe quel modèle
2. Gère facilement les mots hors du vocabulaire