Communication Langagière

Fabien Ringeval, basé sur le cours M2 IDL (F. Portet)

Polytech Grenoble

February 3, 2022

Content

Architecture Encodeur/Décodeur

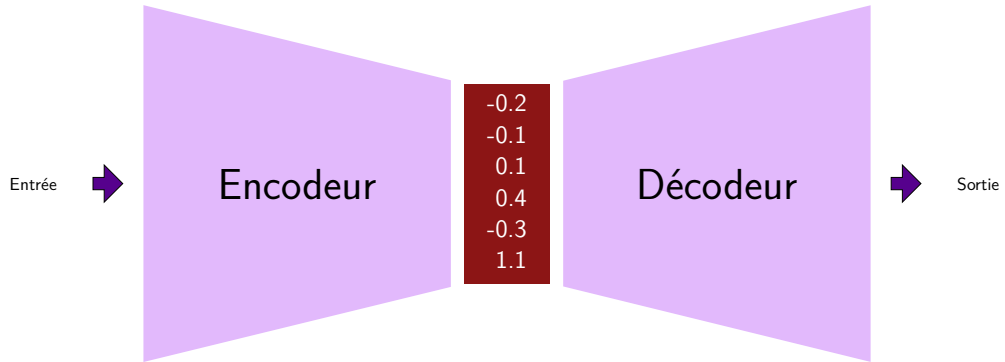
Architecture Encodeur/Décodeur

Stratégies de décodage

Architecture encodeur/décodeur avec attention

Architecture Encodeur-Décodeur

Architecture introduite dans le TALN par la traduction automatique [Kalchbrenner et Blunsom, 2013, Sutskever et coll., 2014].



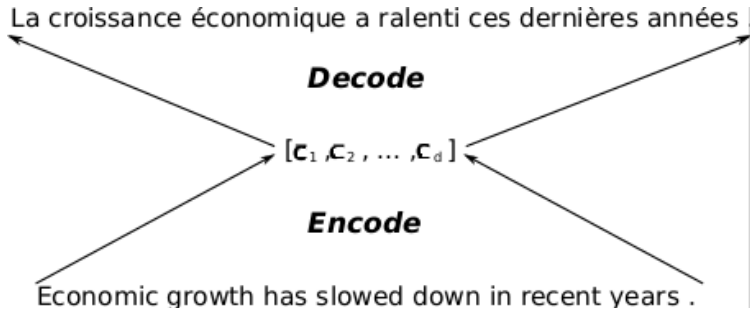
L'encoder extrait un vecteur de représentation de taille fixe (le contexte) à partir d'une entrée de taille quelconque. À partir de cette représentation de taille fixe, le décodeur génère une sortie de taille quelconque

Rupture avec les anciens modèles de traduction automatique statistique :

- les réseaux de neurones n'utilisent qu'une fraction de la mémoire des modèles statistiques
- possibilité d'entraînement joint de tous les composants du modèle

Architecture Encodeur-Décodeur pour le Seq2seq

Tâche de séquence à séquence (seq2seq) [Sutskever et coll., 2014, Cho et coll., 2014a]. Revient formellement à apprendre la probabilité conditionnelle $p(y|x)$ d'une séquence cible y sachant une séquence source x.



tiré de [Cho et coll., 2014b]

Encodeur

Construit c une représentation continue et de taille fixe de la séquence d'entrée

On utilise généralement : (bi)RNN, (bi)LSTM, (bi)GRU mais également CNN, Graph ou transformer.

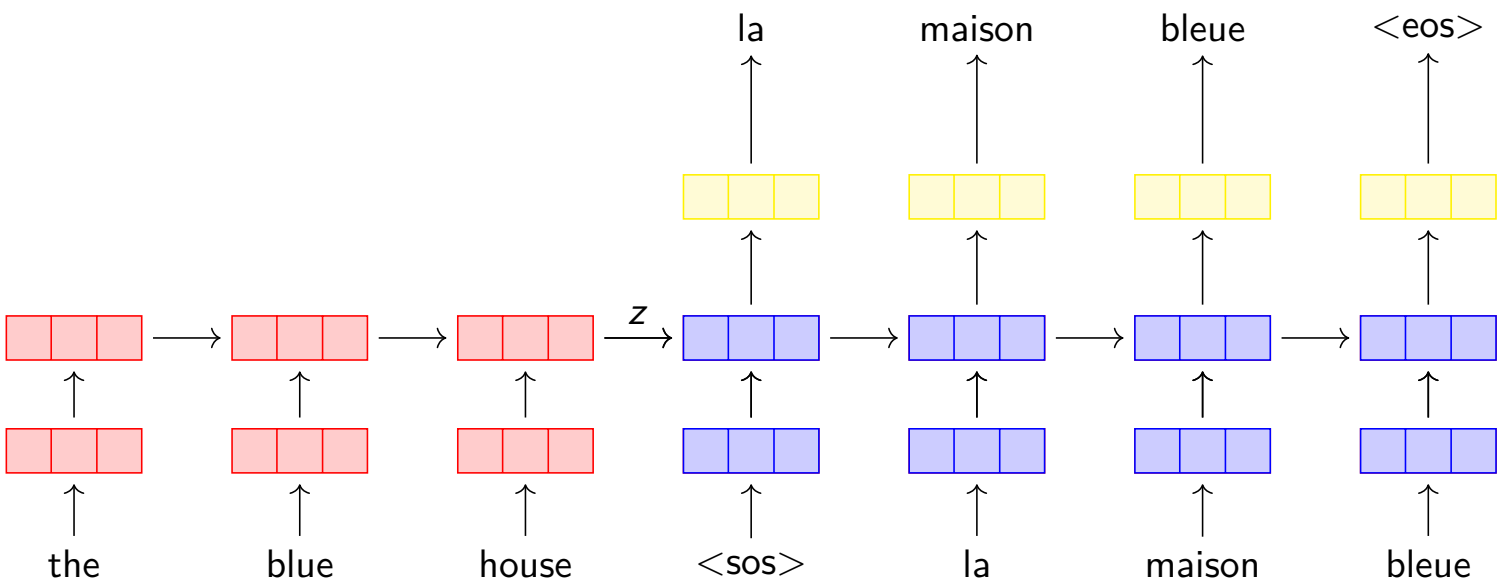
Décodeur

Le décodeur génère une séquence un mot à la fois.

Cette génération est conditionnée par c.

On utilise généralement : (bi)RNN, (bi)LSTM, (bi)GRU ou transformer

Encodeur récurrent



Encodage

$$h_t = \tanh\left(U \times h_{t-1} + V \times x_t\right)$$

Le modèle est entrainé pour représenter:

$$p(x_{t+1}|x_t, \ldots, x_1)$$

- x_t sont des vecteurs représentant les tokens d'entrée.
- À chaque étape, l'encodeur produit un nouveau vecteur h_t (état caché) qui représente le contenu de la chaîne de tokens précédente.
- Le dernier état h_t représente c qui abstrait l'ensemble de l'entrée.
- U et V sont les paramètres appris durant l'apprentissage.
- *tanh* est une fonction non linéaire.

Decodage

$$s_t = \tanh\left(U \times s_{t-1} + V \times y_{t-1}\right)$$

$$y_t = softmax(W \times s_t)$$

Le modèle est entrainé pour représenter:

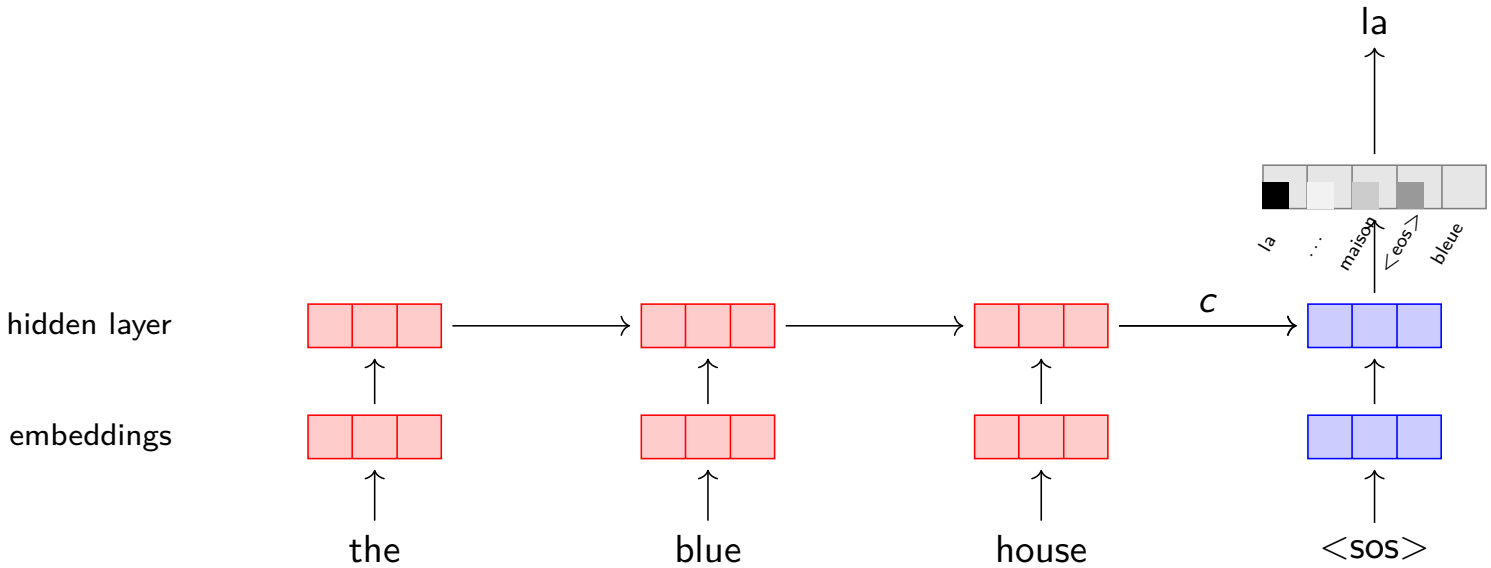
$$p(y_t | y_{t-1}, \ldots, y_1, c)$$

- y_t est le mot prédit à l'instant t.
- s_t est l'état caché du réseau à l'instant t.
- Chaque nouvel état est calculé en tenant compte de l'état précédent s_{t-1} et du dernier mot prédit y_{t-1}.
- La fonction softmax transforme un vecteur de scores en une distribution de probabilités sur le vocabulaire.
- À chaque instant t, le mot prédit y_t est échantillonné à partir de cette distribution de probabilités.

Décodage – 1

Probabilité conditionnelle

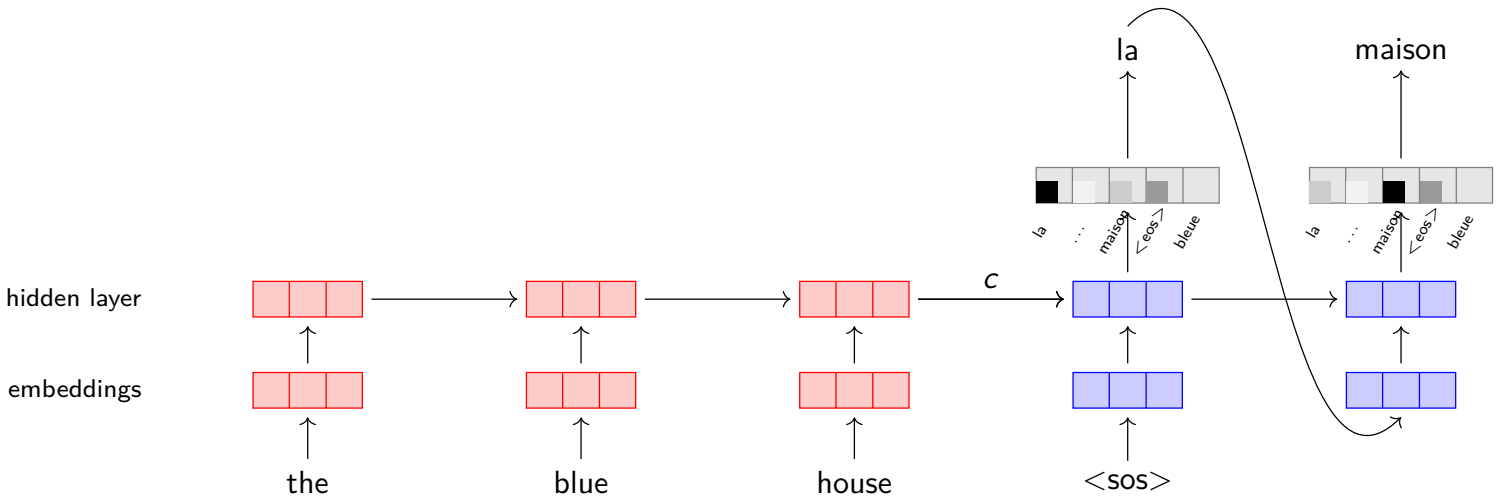
$$p(la| < sos >, house, blue, the)$$



Décodage – 2

Probabilité conditionnelle

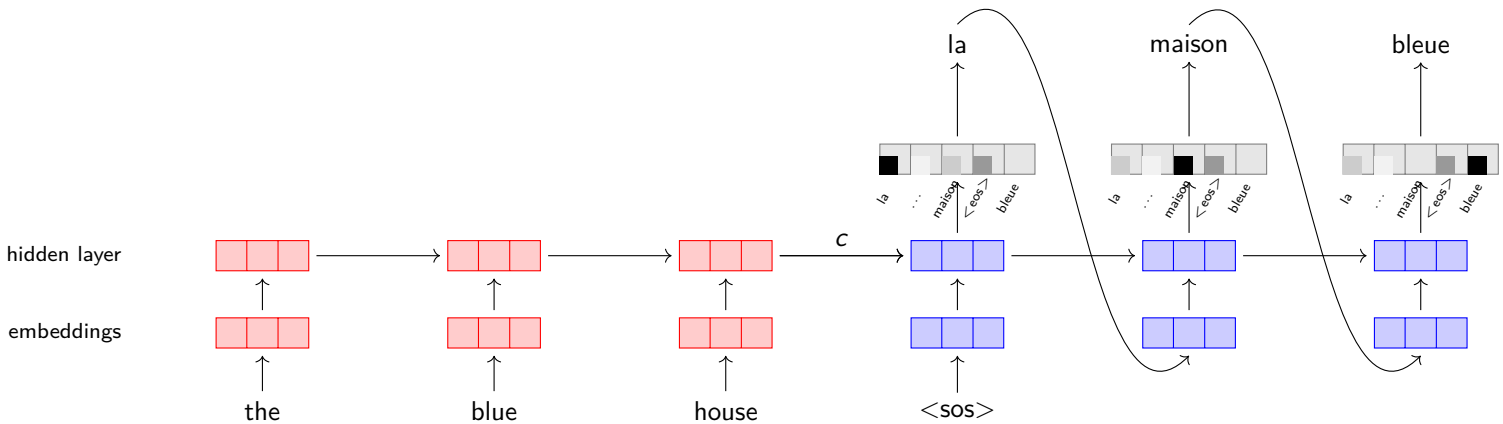
$$p(maison|la, < sos >, house, blue, the)$$



Décodage – 3

Probabilité conditionnelle

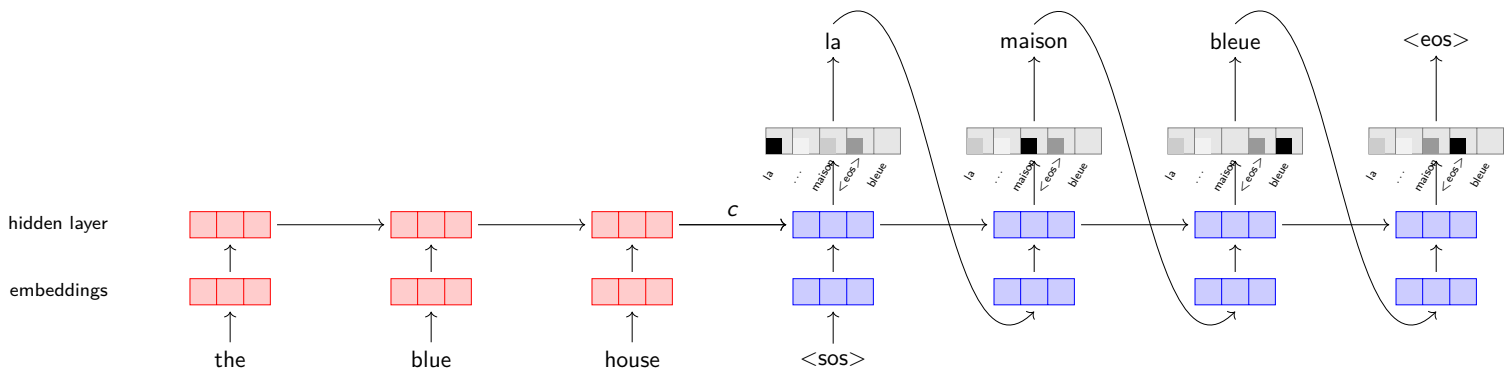
$$p(bleue|maison, la, < sos >, house, blue, the)$$



Décodage – 4

Probabilité conditionnelle

$$p(< eos > | bleue, maison, la, < sos >, house, blue, the)$$



Stratégies de décodage

Architecture Encodeur/Décodeur

Stratégies de décodage

Architecture encodeur/décodeur avec attention

Stratégie de décodage – gready

La technique la plus simple pour trouver la meilleure séquence de sortie et la méthode gloutonne (*gready*).

À chaque étape choisir le terme de plus grande probabilité

...	...
la	.2
...	...
maison	.1
...	...
bleue	.01
...	...
grange	.01
...	...
sorcière	.01
...	...

étape 1

...	...
la	.01
...	...
maison	.4
...	...
bleue	.01
...	...
grange	.01
...	...
sorcière	.01
...	...

étape 2

...	...
la	.01
...	...
maison	.3
...	...
bleue	.67
...	...
grange	.01
...	...
sorcière	.01
...	...

étape 3

$\Rightarrow$ *La maison bleue*

Limites :

- décision locale
- sujet à répétition $\rightarrow$ *La maison La maison La maison La maison La maison La maison La maison*

Stratégie de décodage – *beam search*

Une technique plus sophistiquée est la recherche par faisceaux (*beam search*). À chaque étape, on garde les N meilleures hypothèses et on relance le calcul.
Comment calculer la meilleure hypothèse ?

$$p(y_{1:T}|c) = \prod_{t=1}^{T} p(y_t|y_{1:t-1}, c)$$

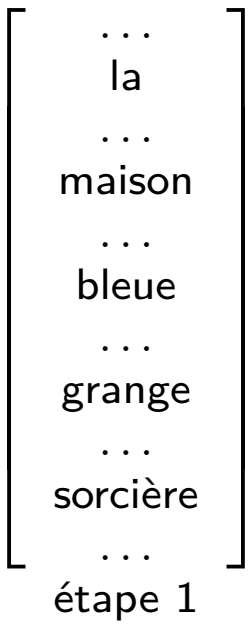
Exemple :

$$p(y_2, y_1|c) = p(y_1|c) \times p(y_2|y_1, c)$$

Stratégie de décodage – *beam search*

Une technique plus sophistiquée est la recherche par faisceaux (*beam search*). À chaque étape, on garde les N meilleures hypothèses et on relance le calcul.

Exemple avec $N = 3$



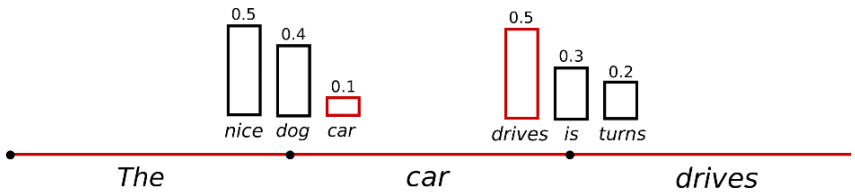
Limites :

- Technique très coûteuse!
- Peut également amener des répétitions
- hypothèses alternatives souvent très proches.

Stratégie de décodage – tirage aléatoire

Tirer aléatoirement le terme selon la distribution donnée par le vecteur de sortie (*sampling*).

- Évite les répétitions
- Ajoute de la nouveauté
- Focalise le sampling sur les termes les plus probables.



tiré de `https://huggingface.co`

Autres techniques de *sampling* :

- $Top - K$, redistribuer la densité de probabilité sur les k termes les plus probables [Fan et coll., 2018].
- $Top - p$ (nucleus), redistribuer la densité de probabilité sur les termes les plus probables dont la somme de probabilités est $\geq p$ [Holtzman et coll., 2020].

Architecture encodeur/décodeur avec attention

Architecture Encodeur/Décodeur

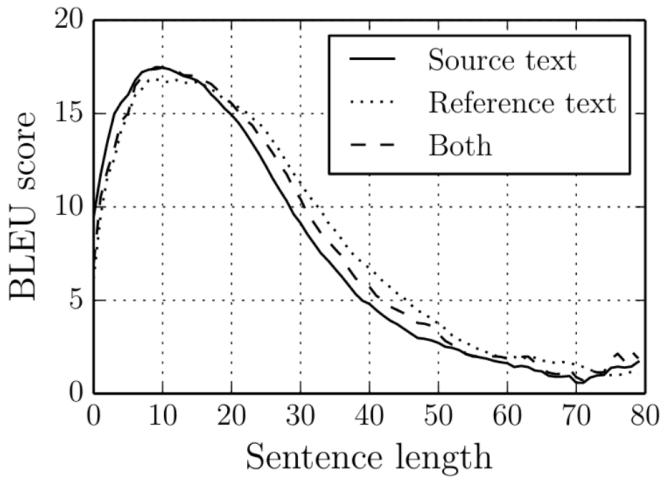
Stratégies de décodage

Architecture encodeur/décodeur avec attention

Attention

Limites de l'architecture Encodeur-Décodeur

- compresse toutes les informations nécessaires d'une séquence source dans un vecteur de longueur fixe sans retour à la source
- les performances se détériorent rapidement à mesure que la longueur d'une séquence source augmente



tiré de [Cho et coll., 2014b]

Mécanisme d'attention pour compléter le modèle Encodeur/Décodeur [Bahdanau et coll., 2016].

- Ce mécanisme recherche un ensemble de positions dans la séquence source où l'information la plus pertinente est concentrée.
- Le modèle prédit ensuite un mot cible sur la base des vecteurs de contexte associés à ces positions sources et tous les mots cibles générés précédemment

Décodage

Sans attention :

$$p(\mathbf{y}) = \prod_{t=1}^{T} p(y_t \mid y_1, \cdots, y_{t-1}, \mathbf{x}), \tag{1}$$

Où $\mathbf{y} = \left(y_1, \cdots, y_{T_y}\right)$.
Les probabilités conditionnelles sont modélisées par

$$p(y_t \mid y_1, \cdots, y_{t-1}, \mathbf{x}) = g(y_{t-1}, s_t, c), \tag{2}$$

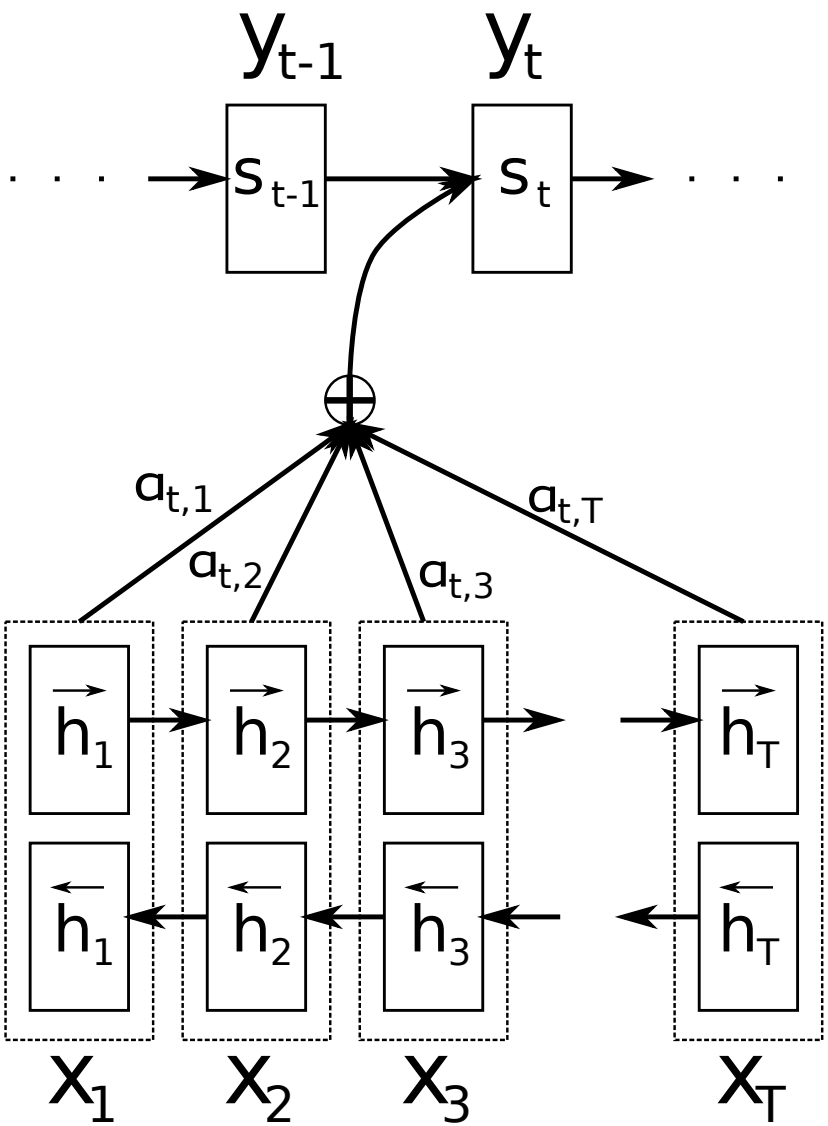
Où g est une fonction (réseau de neurones) qui estime y_t, et s_t est l'état caché du décodeur.
Avec Attention : le vecteur contexte c_i est modifié à chaque étape de décodage

$$p(y_i | y_1, \ldots, y_{i-1}, \mathbf{x}) = g(y_{i-1}, s_i, c_i), \tag{3}$$

Où s_i est un état caché calculé à chaque instant i

$$s_i = f(s_{i-1}, y_{i-1}, c_i).$$

Décodage avec attention



tiré de [Bahdanau et coll., 2016]

c_i est une somme pondérée:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j. \qquad (4)$$

Où α_{ij} est

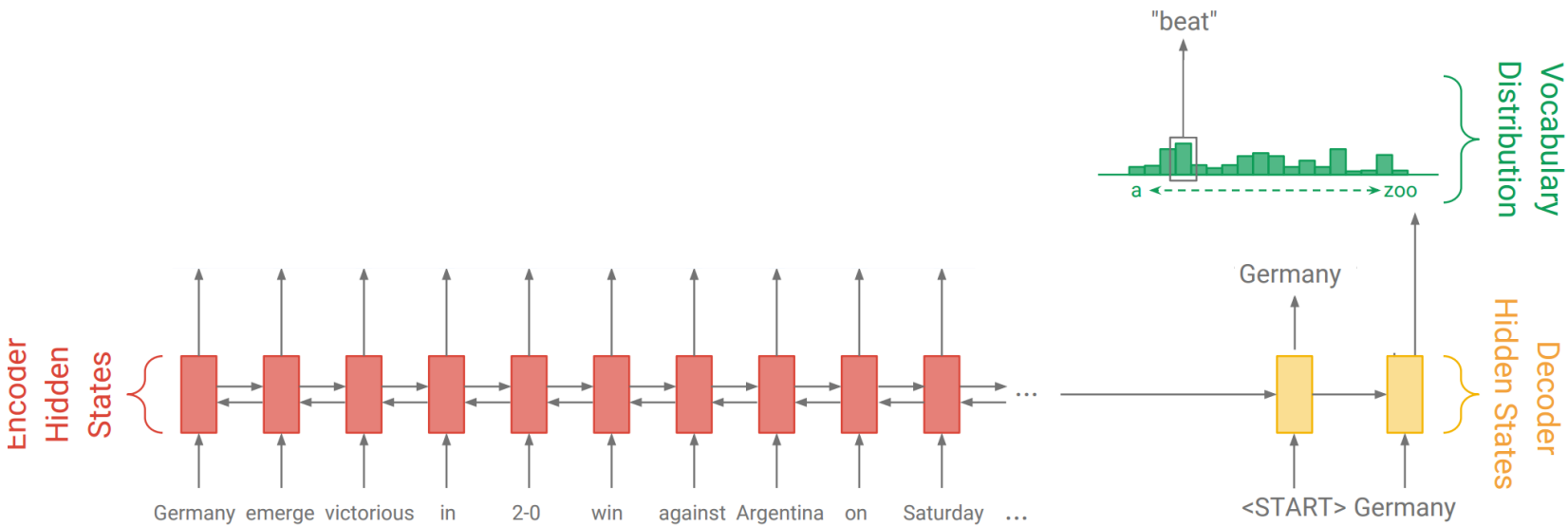
$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}, \qquad (5)$$

et

$$e_{ij} = a(s_{i-1}, h_j)$$

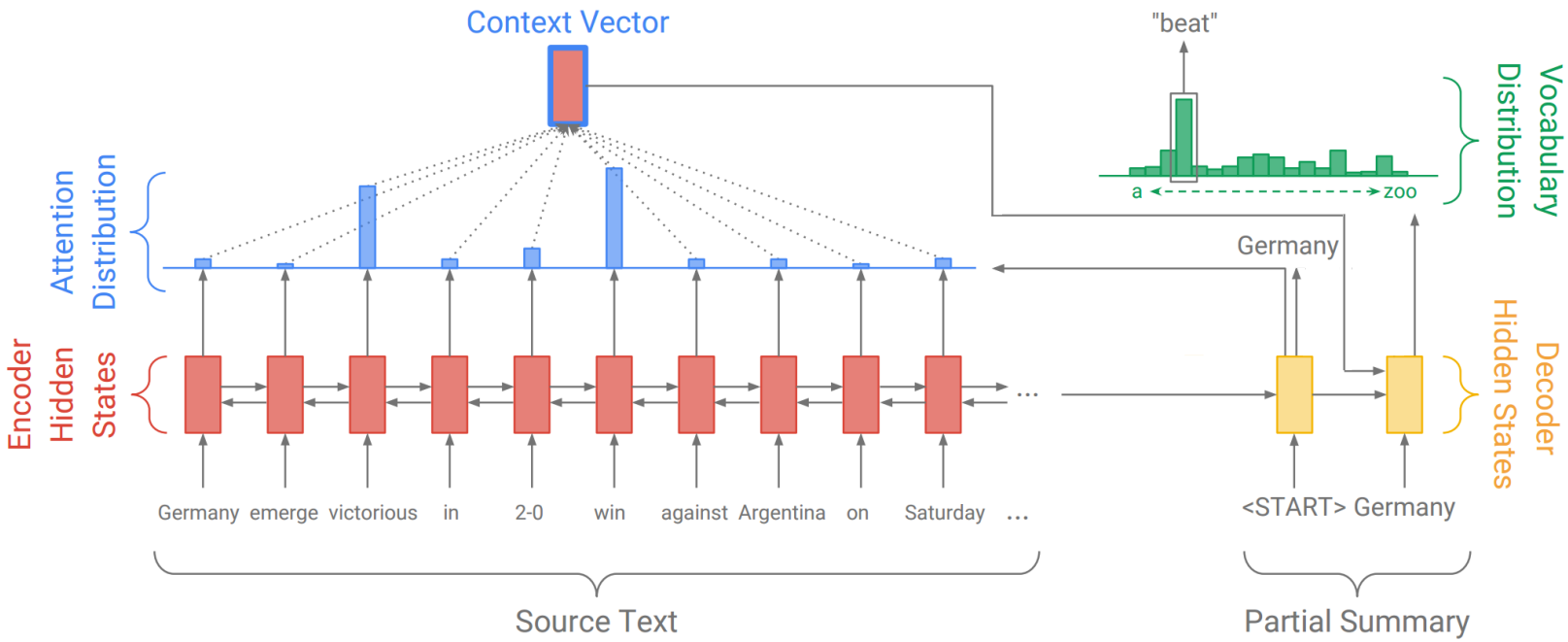
a est un simple réseau de neurones dont les paramètres sont appris conjointement avec le modèle encodeur décodeur.

seq2seq sans attention



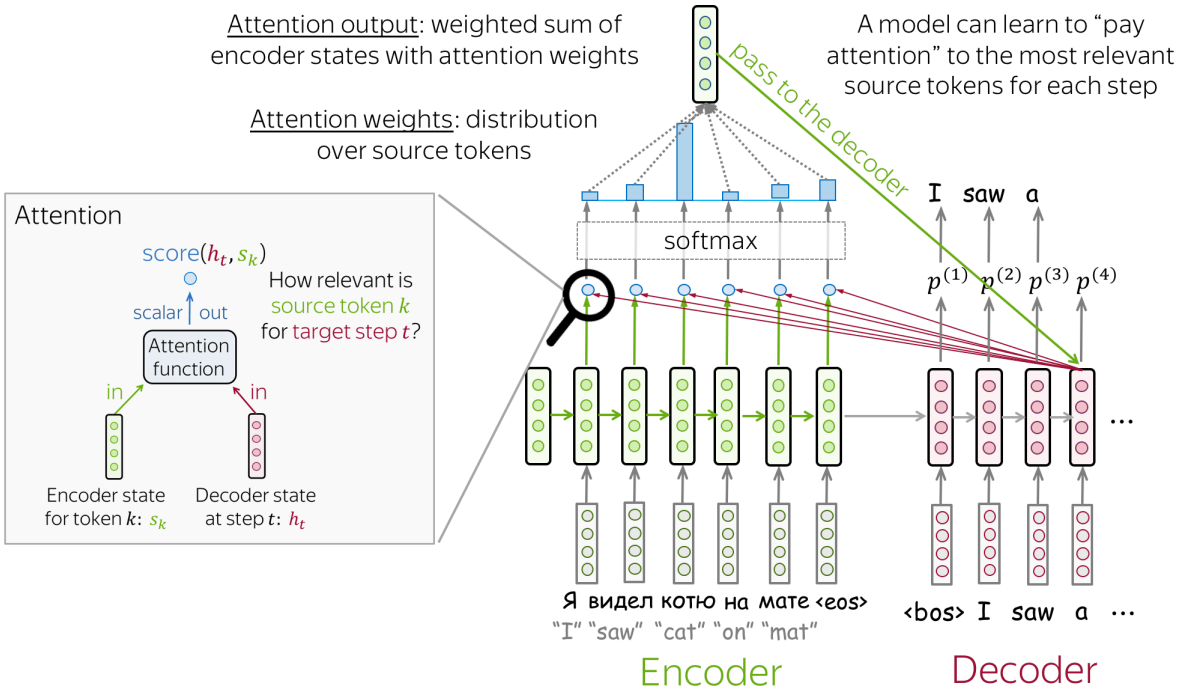
adapté de `http://www.abigailsee.com`

seq2seq avec attention



adapté de `http://www.abigailsee.com`

Avec attention – détails



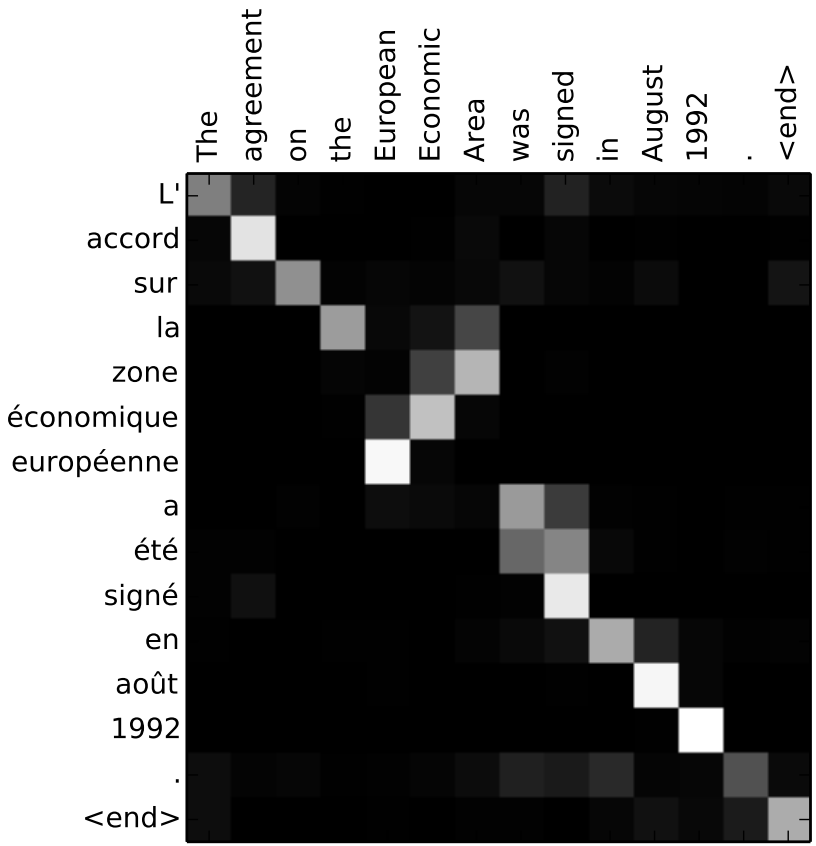
tiré de `https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html`

Plusieurs facon de calculer l'attention

- Dot-product (produit scalaire) : $\mathbf{e}_{ij} = \mathbf{s}_{i-1}^T \mathbf{h}_j$
- Bi-linear ("Luong attention"[Luong et coll., 2015]) : $\mathbf{e}_{ij} = \mathbf{s}_{i-1}^T \mathbf{W} \mathbf{h}_j$
- MLP ("Bahdanau attention"[Bahdanau et coll., 2016]) : $\mathbf{e}_{ij} = MLP(\mathbf{s}_{i-1}, \mathbf{h}_j)$

Attention pour analyser le comportement du réseau

Traduction Anglais vers Français



Bilan

Encoder/décodeur

- apprentissage de bout-en-bout
- force l'abstraction (via le *context vector*)

- défi des dépendances long termes
- représentation du contexte

- l'encodage bi-directionnel permet une meilleure représentation de l'entrée
- l'attention permet de combattre le goulot étranglement du *context vector*
- $\Rightarrow$ Utilisation plus intensive de l'attention.

References I

Bahdanau, D., Cho, K. et Bengio, Y. (2016).
Neural Machine Translation by Jointly Learning to Align and Translate.
arXiv:1409.0473 [cs, stat].
arXiv: 1409.0473.

Cho, K., Merrienboer, B., Gulcehre, C., Bougares, F., Schwenk, H. et Bengio, Y. (2014a).
Learning phrase representations using rnn encoder-decoder for statistical machine translation.
Dans *EMNLP*.

Cho, K., van Merrienboer, B., Bahdanau, D. et Bengio, Y. (2014b).
On the Properties of Neural Machine Translation: Encoder-Decoder Approaches.
arXiv:1409.1259 [cs, stat].
arXiv: 1409.1259.

Fan, A., Lewis, M. et Dauphin, Y. (2018).
Hierarchical Neural Story Generation.
arXiv:1805.04833 [cs].
arXiv: 1805.04833.

Holtzman, A., Buys, J., Du, L., Forbes, M. et Choi, Y. (2020).
The Curious Case of Neural Text Degeneration.
arXiv:1904.09751 [cs].
arXiv: 1904.09751.

References II

Kalchbrenner, N. et Blunsom, P. (2013).
Recurrent continuous translation models.
Dans *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1700–1709.

Luong, T., Pham, H. et Manning, C. D. (2015).
Effective approaches to attention-based neural machine translation.
Dans *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Sutskever, I., Vinyals, O. et Le, Q. V. (2014).
Sequence to sequence learning with neural networks.
Dans *Advances in neural information processing systems*, pages 3104–3112.