Cours de Traitement Automatique du Langage

Chapitre 04 : Etiquetage morpho-syntaxique

DI5 – 2021-22

Polytech ’Tours J.Y. RAMEL

Introduction

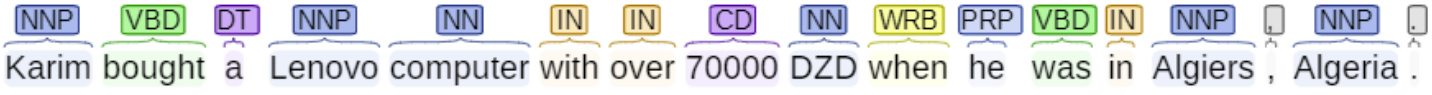


FIGURE – Un exemple d'étiquetage morpho-syntaxique [https://corenlp.run/]

- Ou sont les noms, les verbes, les pronoms, etc.
  - can: (1) pouvoir (2) mettre en boıte (3) boıte
  - will: (1) exprime le futur (2) un nom propre masculin (3) vouloir (4) testament
- Le dictionnaire ne suffit pas pour déterminer l'étiquette / la classe grammaticale d'un mot…

Exemple d'une phrase en anglais

We can can the can
Will Will will the will to Will?

- Applications similaires (exploitant `https://corenlp.run/`)

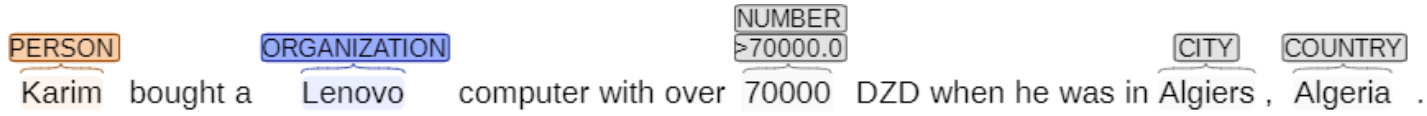


FIGURE – Un exemple de la reconnaissance d'entités nommées [https://corenlp.run/]

Etiquetage des séquences

Objectifs

- Définir les catégories grammaticales de chaque mot dans une phrase (POS)
  - Nature, catégorie grammatical, catégorie lexicale, classe grammaticale, espèce grammaticale, partie du discours (**Part Of Speech**)

Formulation du problème

- $w = w_1, \ldots, w_n$: une séquence d'entrée
- $t = t_1, \ldots, t_n$: une séquence des étiquettes

$$\arg\max_t P(t|w)$$

- **Modèle génératif** $\arg\max_t P(t|w) = \arg\max_t P(t)P(w|t)$
- **Modèle discriminatif** calculer $\arg\max_t P(t|w)$ directement

- Nécessite d'annoter des corpus
  - Annoter un texte manuellement pour le test
  - Un autre pour l'entraınement si on utilise l'apprentissage automatique

Nécessité de corpus annotés

- Avec des classes universelles ?

https://universaldependencies.org/u/pos/

Classe ouverte	Classe fermée	Autres
ADJ : adjectif	***ADP*** : adposition	***PUNCT*** : ponctuation
ADV : adverbe	***AUX*** : auxiliaire	***SYM*** : symbole
INTJ : interjection	***CCONJ*** : conjonction de coordination	***X*** : Autres
NOUN : nom	***DET*** : déterminant	
PROPN : nom propre	***NUM*** : numérique	
VERB : verbe	***PART*** : particule	
	PRON : pronom	
	SCONJ : conjonction de subordination	

- Universal TreeBank [Petrov et al., 2012]
  - Projet ouvert visant a fournir des corpus annotés pour plusieurs langues
  - En utilisant les mêmes classes grammaticales

https://universaldependencies.org/

```
Battle-tested/JJ Japanese/JJ industrial/JJ managers/NNS
here/RB always/RB buck/VBP up/RP nervous/JJ newcomers/NNS
with/IN the/DT tale/NN of/IN the/DT first/JJ of/IN
their/PP$ countrymen/NNS to/TO visit/VB Mexico/NNP ,/,
a/DT boatload/NN of/IN samurai/FW warriors/NNS blown/VBN
ashore/RB 375/CD years/NNS ago/RB ./.
```

FIGURE – Exemple d'un texte annoté de Penn TreeBank [Taylor et al., 2003]

Etiquetage des séquences

Quelles approches ?

- Règles définies manuellement

- Apprentissage automatique
  - Apprendre les règles : Transformation-Based Learning (TBL)
  - Modèles statistiques / probabilistes : Modèles de Markov cachés, champs aléatoires conditionnels (CRF), …
  - Réseaux de neurones

Etiquetage des sequences par HMM

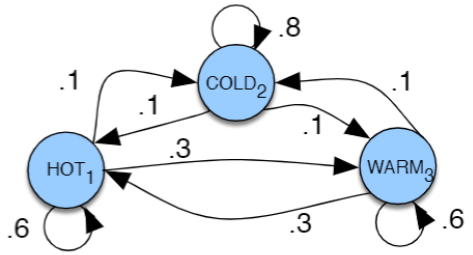
Modèles de Markov Cachés (HMM)

Hypothèse de Markov = $P(q_i=a|q_1,...,q_{i-1}) \approx P(q_i=a|q_{i-1})$

Le futur ne dépend que du présent, pas du passé

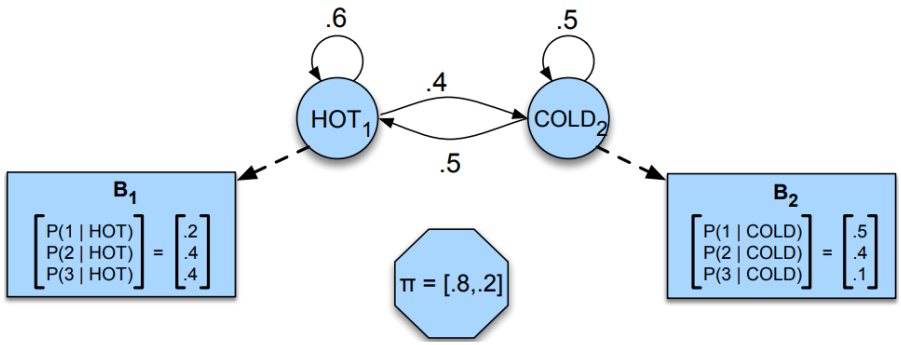
$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}$$

- $Q=\{q_1,q_2,...,q_n\}$: états
- A= matrice des probabilités de transition - $\sum_j a_{ij}=1, \forall i$
- $\pi= [\pi_1,\pi_2,...,\pi_n]$: distribution initiale des probabilités des états - $\sum_i \pi_i=1$
- $O=o_1\ o_2\ ...\ o_T$: séquence des événements observés (mots)
- $B=b_i(o_t)$: probabilités d'observation (probabilités d'émission), chacune représente la probabilité de génération d'une observation o_t dans un état q_i
- Les états cachés sont les tags/étiquettes (*parts of speech*) t_i à prédire pour chaque observation (*mots observés*)
- Independence des observations (la probabilité de l'observation o_i dépend uniquement de l'état qui l'a produit q_i) :
- $P(o_i|q_1\ ...q_i,...,q_T\ ,o_1,...,o_i,...,o_T\ ) = P(o_i|q_i)$



- En TAL, 1 HMM = 1 modèle de bi-grams

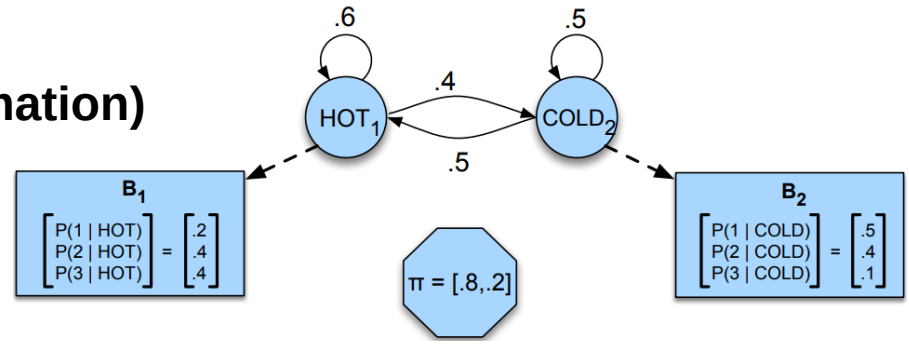
HMM ice-cream eating



Etiquetage des sequences par HMM

Modeles de Markov Cachés (HMM)

Estimation des vraisemblances (likelihood estimation)



- Connaissant la HMM λ = (A,B) et la séquence d'observations O, déterminer P(O|λ).

- Par exemple, connaissant la HMM ice-cream eating, quelle est la probabilité de la séquence 3 1 3
  cas réel = sans avoir connaissance des états caches associés (Hot/Cold) ?

- Facile quand on connait la séquence O
  P(3 1 3|hot hot cold) = P(3|hot)×P(1|hot)×P(3|cold)

$$P(O|Q) = \prod_{i=1}^{T} P(o_i|q_i)$$

- Moins facile quand on ne connait pas O
  P(O) = somme de toutes les séquences d'états possibles, pondérées par leur probabilité
  P(3 1 3) = P(3 1 3, cold cold cold) + P(3 1 3, cold cold hot) + P(3 1 3,hot hot cold) +...

$$P(O) = \sum P(O,Q) = \sum P(O|Q)P(Q)$$

avec

$$P(O,Q) = P(O|Q) \times P(Q) = \prod_{i=1}^{T} P(o_i|q_i) \times \prod_{i=1}^{T} P(q_i|q_{i-1})$$

- En pratique le nb d'états possibles N^T est très grand.
- Utilisation de l'algorithme FORWARD (prog. dynamique) pour un calcul plus rapide par génération d'un treillis
- Propagation des probabilités forward (p d'être dans l'état j après avoir vu les t premières observations, étant donné λ)

$$\alpha_t(j) = P(o_1, o_2 \ldots o_t, q_t = j|\lambda) = \alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t)$$

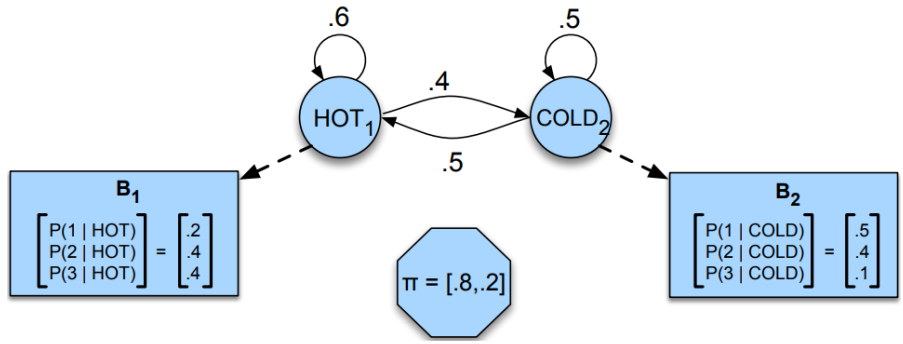
$\alpha_{t-1}(i)$	the **previous forward path probability** from the previous time step
a_{ij}	the **transition probability** from previous state q_i to current state q_j
$b_j(o_t)$	the **state observation likelihood** of the observation symbol o_t given the current state j

Etiquetage des sequences par HMM

Modeles de Markov Cachés (HMM)

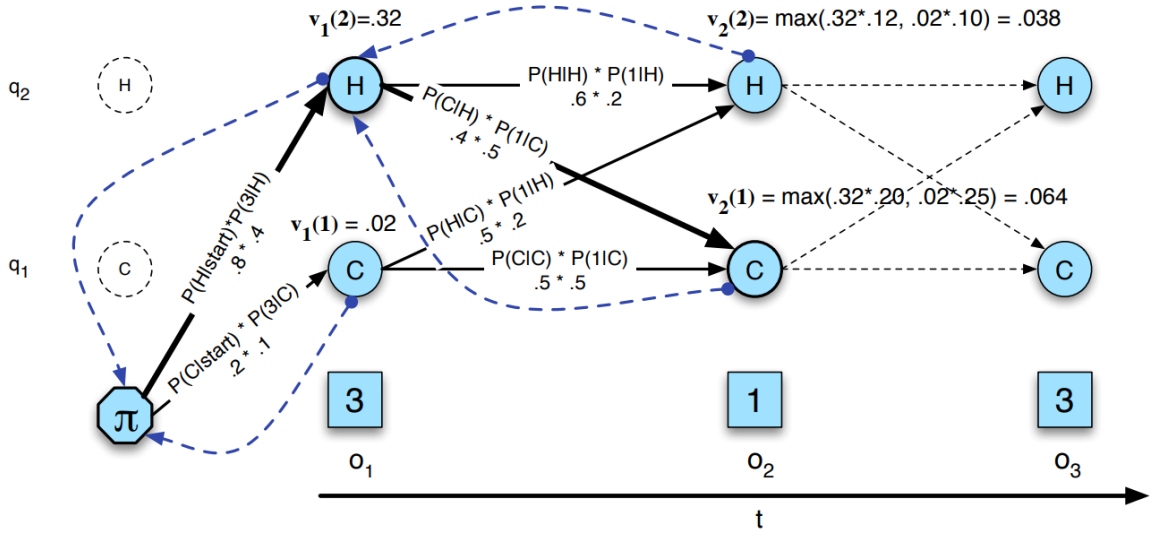
Décodage des tags (Viterbi)

Estimation de la vraisemblance maximum



- Etant donnée une HMM λ = (A,B) et une séquence d'observations O = $o_1,o_2,...,o_T$
- Trouver la séquence d'états Q = $q_1q_2q_3$ ...q_T la plus probable

- Adaptation de l'algorithme forward en remplaçant l'opérateur de Sum() par l'operateur Max() lors du parcours du treillis + mécanisme permettant le backtracking



$v_{t-1}(i)$	the **previous Viterbi path probability** from the previous time step
a_{ij}	the **transition probability** from previous state q_i to current state q_j
$b_j(o_t)$	the **state observation likelihood** of the observation symbol o_t given the current state j

Etiquetage des sequences par HMM

Modèle de Markov Caché (HMM) – Etiquetage (décodage des tags)

- Etant donné un modèle HMM λ= (A,B) et une séquence d'observations/mots (o_i → w_j) : $w=w_1w_2...w_n$
- Estimer la séquence des étiquettes (qj → tj) : $t=t_1,t_2...t_n$
- Algorithme de Viterbi

Viterbi

Data : $w = w_1 \dots w_T$, HMM $\lambda = (A, B)$ avec N états
Result : *meilleur_chemin*, *prob_chemin*
Créer une matrice $viterbi[N, T]$;
pour *état* $s = 1 \dots N$ **faire**
  $viterbi[s, 1] = \pi_s * b_s(w_1)$; $backpointer[s, 1] = 0$;
fin
pour $t = 1 \dots T$ **faire**
  pour *état* $s = 1 \dots N$ **faire**
    $viterbi[s, t] = \max_{s'=1}^{N} viterbi[s', t-1] * a_{s',s} * b_s(w_t)$;
    $backpointer[s, t] = \arg\max_{s'=1}^{N} viterbi[s', t-1] * a_{s',s} * b_s(w_t)$;
  fin
fin
$prob\_chemin = \max_{s=1}^{N} viterbi[s, T]$; $pointeur\_chemin = \arg\max_{s=1}^{N} viterbi[s, T]$;
meilleur_chemin est le chemin qui commence par *pointeur_chemin* et qui suit *backpointer*
return *meilleur_chemin*, *prob_chemin*;

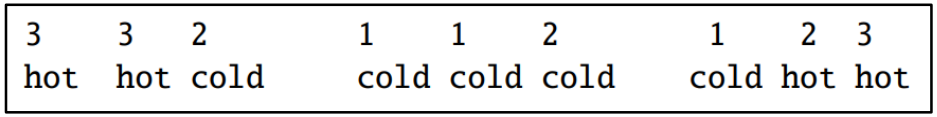
Etiquetage des sequences par HMM

Modèles de Markov Cachés (HMM)

Apprentissage

- En entrée : une séquence **d'observations étiquetées O**
  et un vocabulaire d'états cachés potentiels Q = classes du POS (= hot/cold ici).
- En sortie : λ (A et B)

- **Si on connait les observations et leurs états cachés**



$\pi_h = 1/3 \quad \pi_c = 2/3$

A

$p(hot|hot) = 2/3 \quad p(cold|hot) = 1/3$
$p(cold|cold) = 2/3 \quad p(hot|cold) = 1/3$

B

$P(1|hot) = 0/4 = 0 \quad p(1|cold) = 3/5 = .6$
$P(2|hot) = 1/4 = .25 \quad p(2|cold = 2/5 = .4$
$P(3|hot) = 3/4 = .75 \quad p(3|cold) = 0$

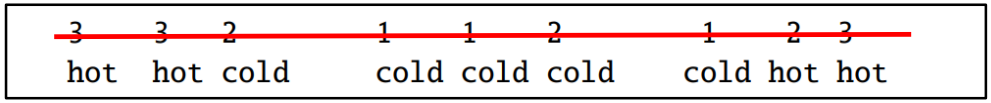
Etiquetage des sequences par HMM

Modeles de Markov Cachés (HMM)

Apprentissage (forward-backward, Baum-Welch algorithm (Baum, 1972), EM algorithm (Dempster, 1977).

- En entrée : une séquence **d'observations non étiquetées O** et un vocabulaire d'états cachés potentiels Q.
- En sortie : λ (A et B)

- Si on ne connait pas les états → Algorithme de FB, Baum-Welch, EM



$\pi_h = 1/3 \quad \pi_c = 2/3$

Â

$p(hot|hot) = 2/3 \quad p(cold|hot) = 1/3$

$p(cold|cold) = 2/3 \quad p(hot|cold) = 1/3$

B = ?

- Estimation itérative (FB) des probabilités dans A et B à partir d'une initialisation de Â et ^B
- Backward probabilité β = probabilité de voir les observations du temps t +1 à la fin, étant dans l'état i au temps t (et étant donné λ)

$\beta_t(i) = P(o_{t+1}, o_{t+2} \ldots o_T \mid q_t = i, \lambda) = \beta_t(i) = \sum_{j=1}^{N} a_{ij}\, b_j(o_{t+1})\, \beta_{t+1}(j), \quad 1 \le i \le N$

function FORWARD-BACKWARD (observations of len T, output vocabulary V, hidden state set Q) returns HMM=(A,B)

initialize A and B

iterate until convergence

E-step

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{\alpha_T(q_F)} \quad \forall\, t \text{ and } j$$

$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\alpha_T(q_F)} \quad \forall\, t,\ i, \text{ and } j$$

M-step

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{k=1}^{N} \xi_t(i,k)}$$

$$\hat{b}_j(v_k) = \frac{\sum_{t=1 s.t.\ O_t = v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

return A, B

Etiquetage des sequences par HMM

- **Mise en œuvre…**

Exemple d'un corpus d'entraînement

- un/DET ordianteur/NOUN peut/VERB vous/PRON aider/VERB
- il/PRON veut/VERB vous/PRON aider/VERB
- il/PRON veut/VERB un/DET ordinateur/NOUN
- il/PRON peut/VERB nager/VERB

Tag → G={DET,NOUN,VERB,PRON}

- Trouver les tags de la phrase : « il peut aider »
  - Entraıner le modèle (seulement sur les mots et les tags nécessaires)
  - Appliquer Viterbi

$$P(VERB|NOUN) = \frac{C(NOUN, VERB)}{C(NOUN)} = \frac{1}{2}$$

$$P(veut|VERB) = \frac{C(VERB, veut)}{C(VERB)} = \frac{2}{7}$$

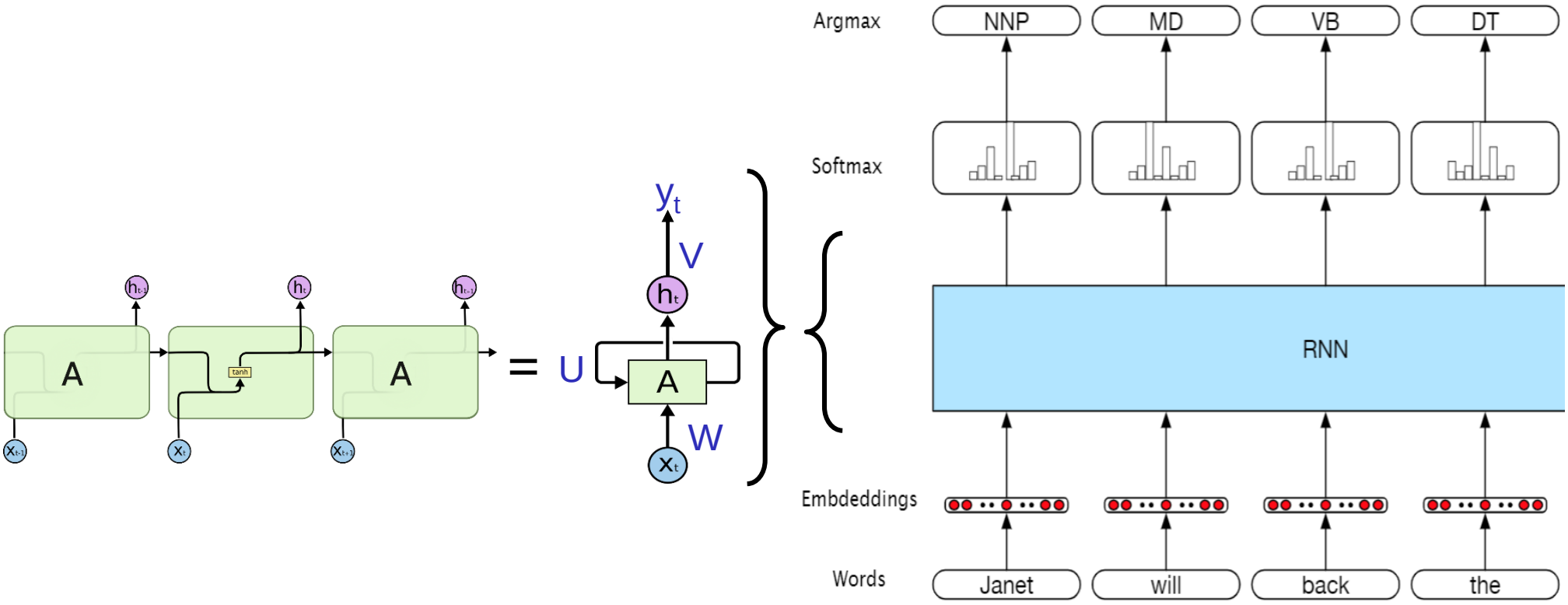
$$\pi_{PRON} = \frac{C(PRON, <s>)}{C(<s>)} = \frac{3}{4}$$

- **Voir tutoriel HMM De Romain Raveaux**

→ http://romain.raveaux.free.fr/document/HMMandNER.html

Etiquetage des sequences par RNN

De manière similaire à la création de ML → RNN

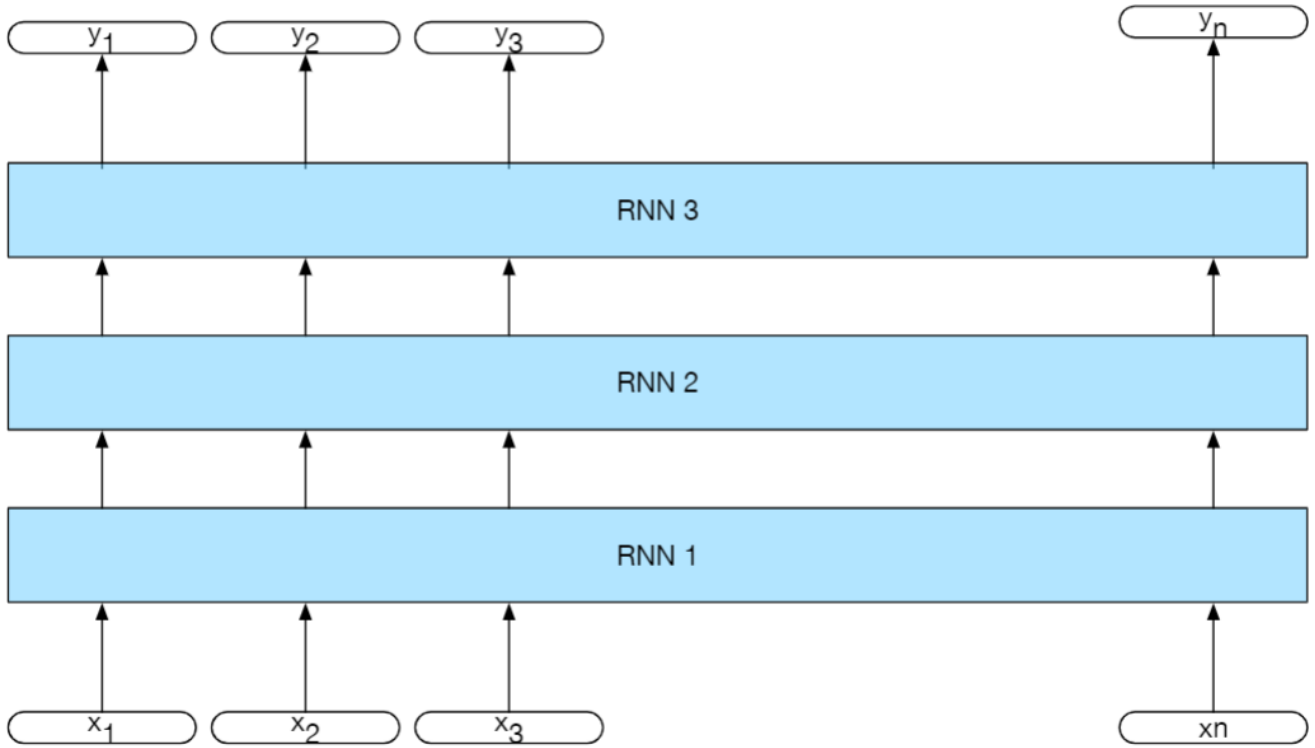


RNN simple pour l'etiquetage morphosyntaxique [Jurafsky and Martin, 2019]

Etiquetage des sequences par RNN

RNN empilés (stacked RNN) « plus deep » …

- Introduction de différents niveaux d'abstraction
- Augmentation du cout d'apprentisage



[Jurafsky and Martin, 2019]

Etiquetage des sequences par RNN

RNN bidirectionnel

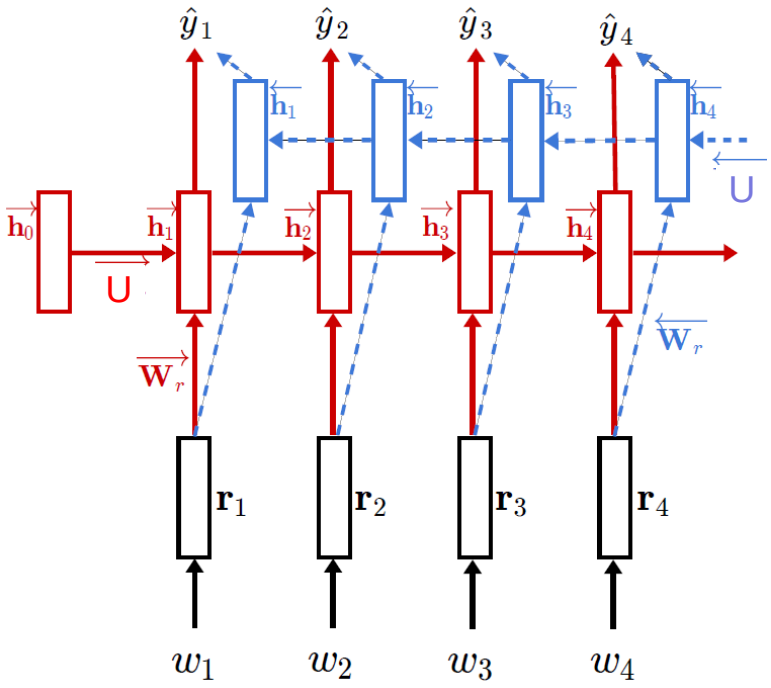
- Dans de nombreuses cas, les séquences complètes s
  disponibles dès le départ (pourtant utilisation des mot
  jusquà t-1 uniquement) !

- Combinaison de 2 RNN :
  - gauche/droite + droite/gauche
  - **Operateur de fusion ⊕ : concat(), sum(), multiply(**

$$h_t^f = RNN_{forward}(x_1^t)$$

$$h_t^b = RNN_{backward}(x_t^n)$$

$$h_t = h_t^f \oplus h_t^b$$



$$\overrightarrow{\mathbf{h}_i} = \phi(\overrightarrow{\mathbf{W}}_r \mathbf{r}_{w_i} + \overrightarrow{\mathsf{U}}\, \overrightarrow{\mathbf{h}_{i-1}})$$

$$\overleftarrow{\mathbf{h}_i} = \phi(\overleftarrow{\mathbf{W}}_r \mathbf{r}_{w_i} + \overleftarrow{\mathsf{U}}\, \overleftarrow{\mathbf{h}_{i+1}})$$

$$\hat{y}_i = f(\mathbf{V} . [\overrightarrow{\mathbf{h}_i} : \overleftarrow{\mathbf{h}_i}])$$

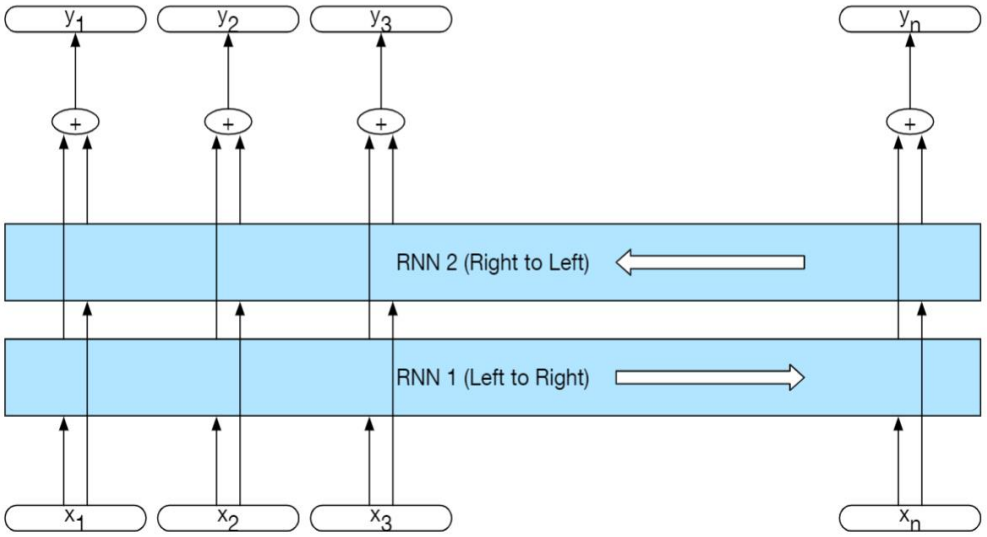
Etiquetage des sequences par RNN

RNN bidirectionnel

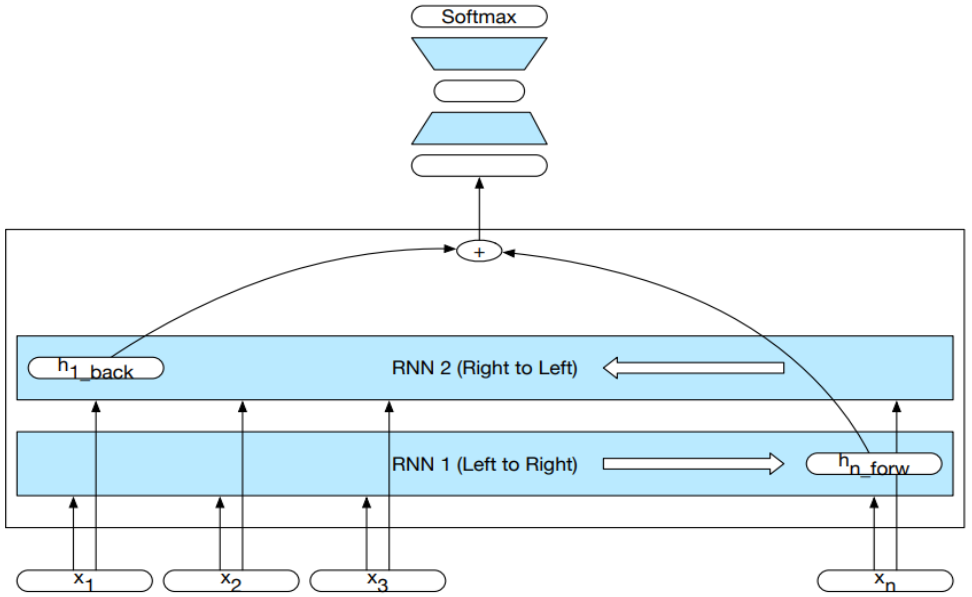
- Pas possible pour la creation de ML

- Mais possible pour :

Ngrams classification (POS)



Sequence classification (using only final states)



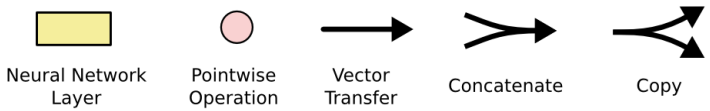
Nouveaux Modèles Neuronaux – LSTM

LSTM Long short-term memory (Hochreiter and Schmidhuber, 1997)

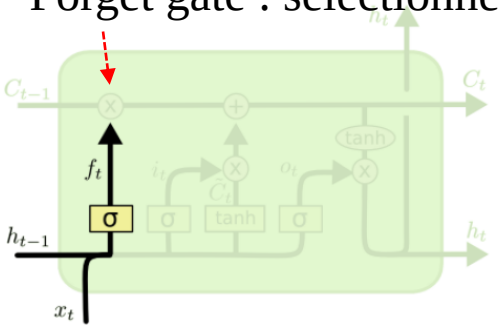
- Pour essayer de mieux exploiter le contexte
  - Supprimer les informations inutiles à long terme
  - Conserver les informations utiles pour les décisions a venir
- En
  - Ajoutant une couche explicitement dédiée à la gestion du contexte capable d'apprendre plutôt que de l'encoder en dur **via une architecture dédiée**
  - Utilisant des unités neuronales **(gates**) pour mieux contrôler le flot d' information au travers des couches → laisser passer (ouvert) ou supprimer l'info (fermer)
  - Gate = 1 couche sigmoïd (0 = fermé, 1 ouvert) et une opération de multiplication
  - **Voir illustration sur slide suivant ➔ Forget gate + add gate + output gate**
- Ca fait bcp de poids à apprendre (dans chaque gate)…

LSTM

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

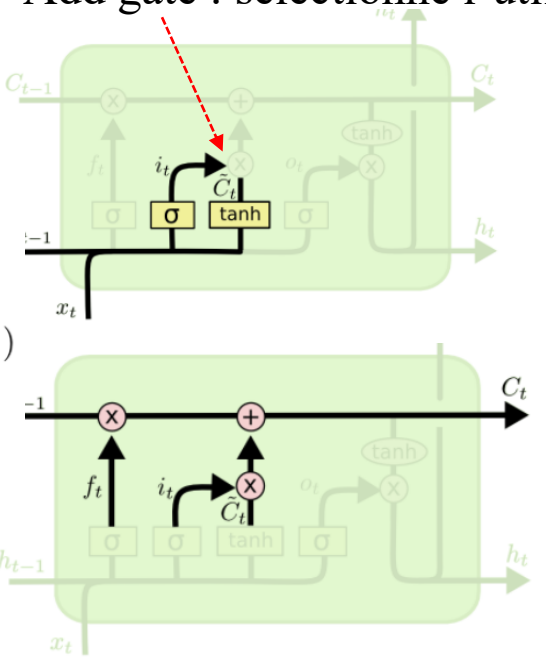


Forget gate : selectionne l'inutile



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Add gate : selectionne l'utile



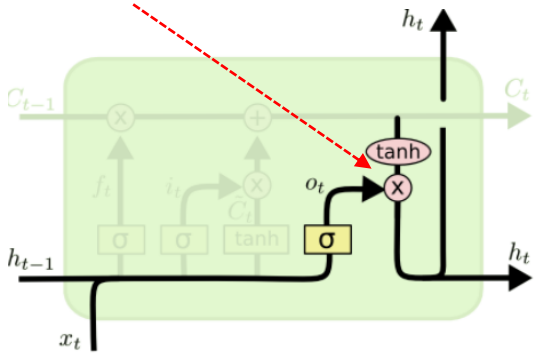
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Met à jour le contexte

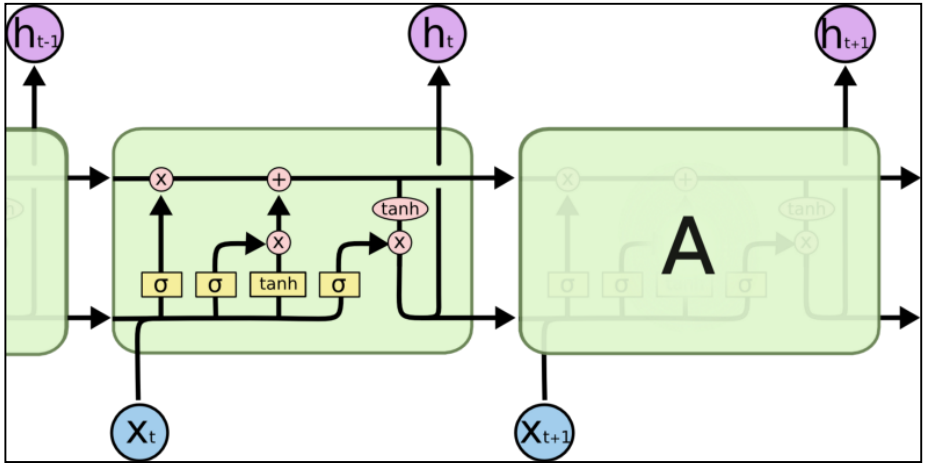
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Output gate : Met à jour l'état ht



$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$



Gated Recurrent Units

GRU : Gated Recurrent Units (Cho et al., 2014)

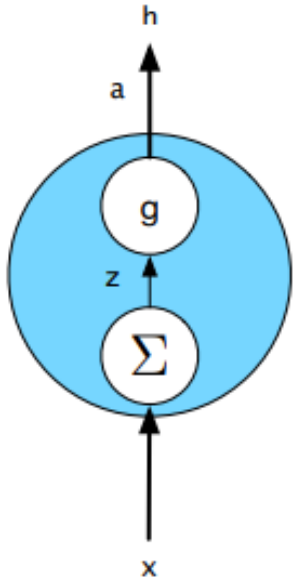
- Reduction du nombre de poids à apprendre
- Que 2 Gates ➔ reset gate, r + update gate, z ➔

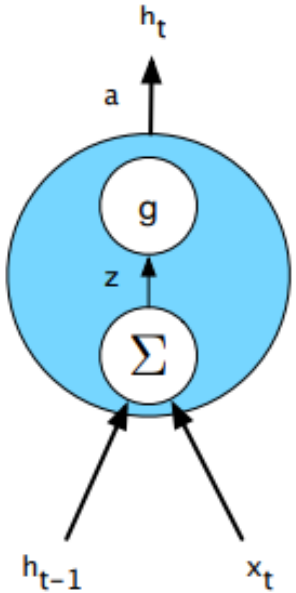
$$r_t = \sigma(U_r h_{t-1} + W_r x_t)$$
$$z_t = \sigma(U_z h_{t-1} + W_z x_t)$$

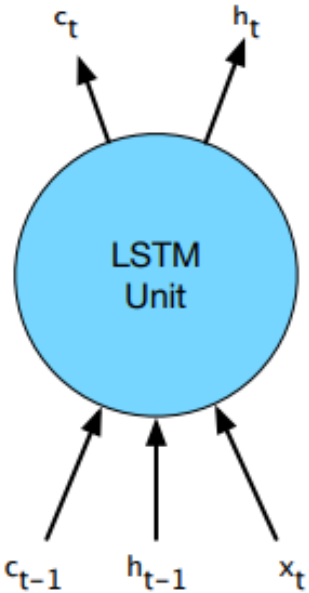
$$\tilde{h}_t = tanh(U(r_t \odot h_{t-1}) + W x_t)$$

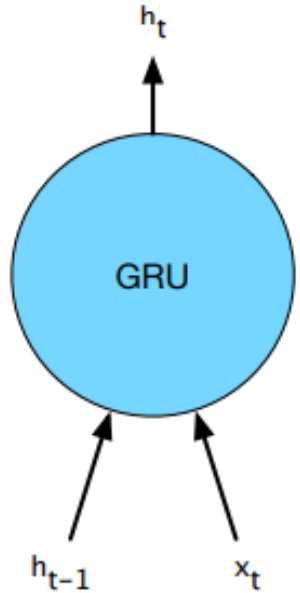
$$h_t = (1 - z_t) h_{t-1} + z_t \tilde{h}_t$$

Résumé ➔









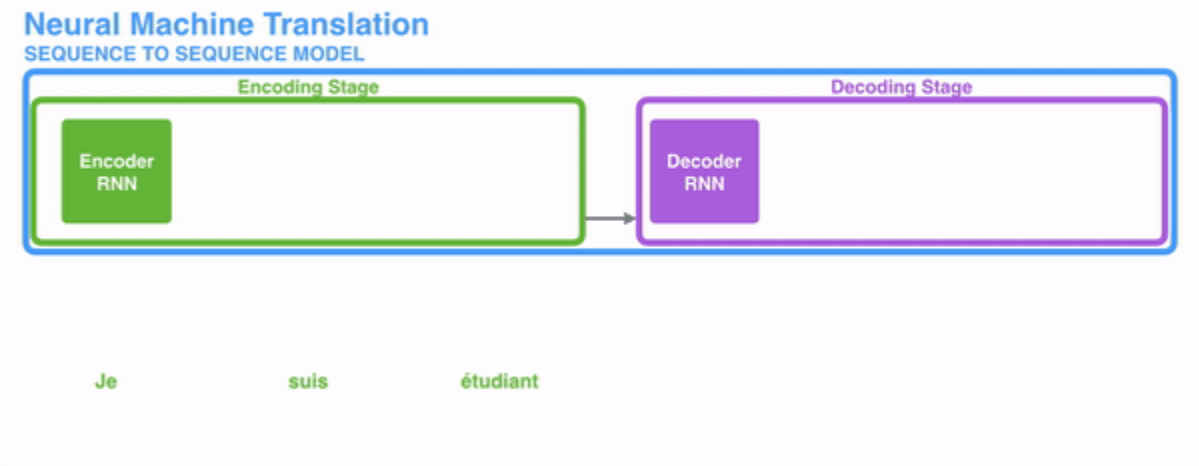
Transformers

https://towardsdatascience.com/transformers-141e32e69591

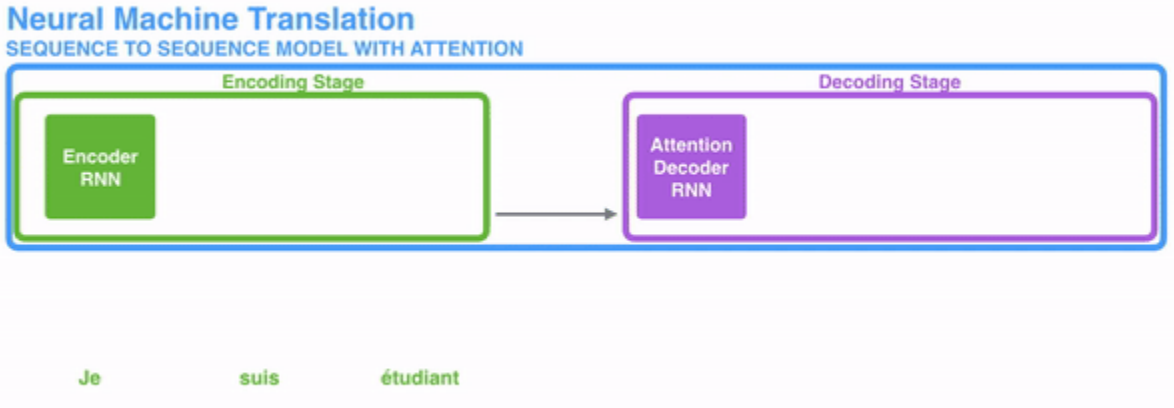
Idées principales

- Elimination les connexions récurrentes→ Back to fully connected networks
- Amélioration du concept de convolutions → Self-Attention
- Exploitation du mécanisme Encodeur – Décodeur

- RNN ➔



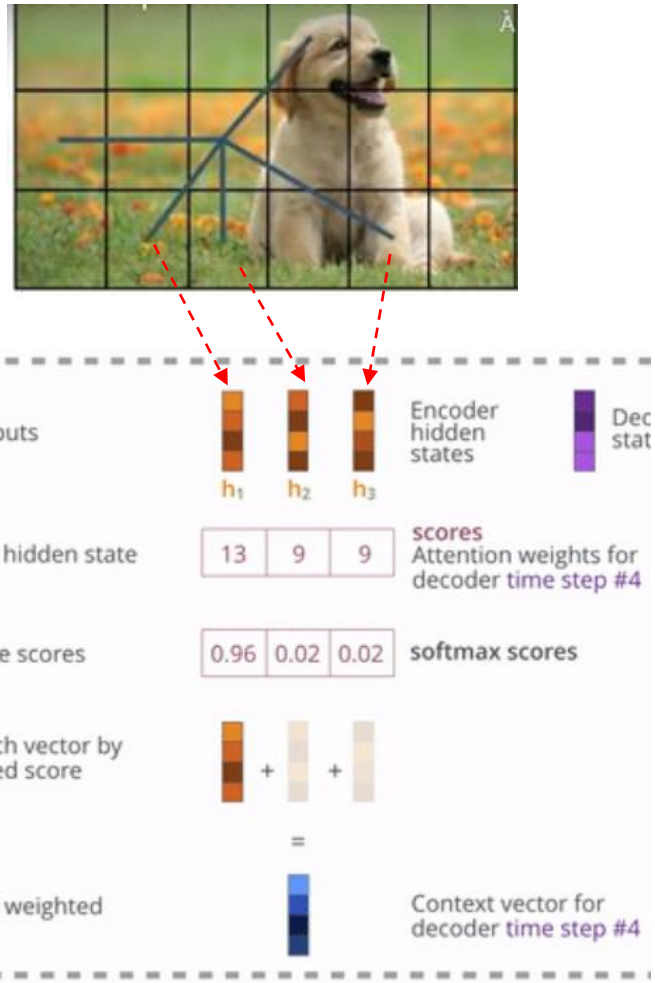
- Enc-Dec ➔



Des CNN aux Transformers

Mécanisme **d'attention**

- Pondération de l'importance relative des éléments les uns par rapport aux autres localement

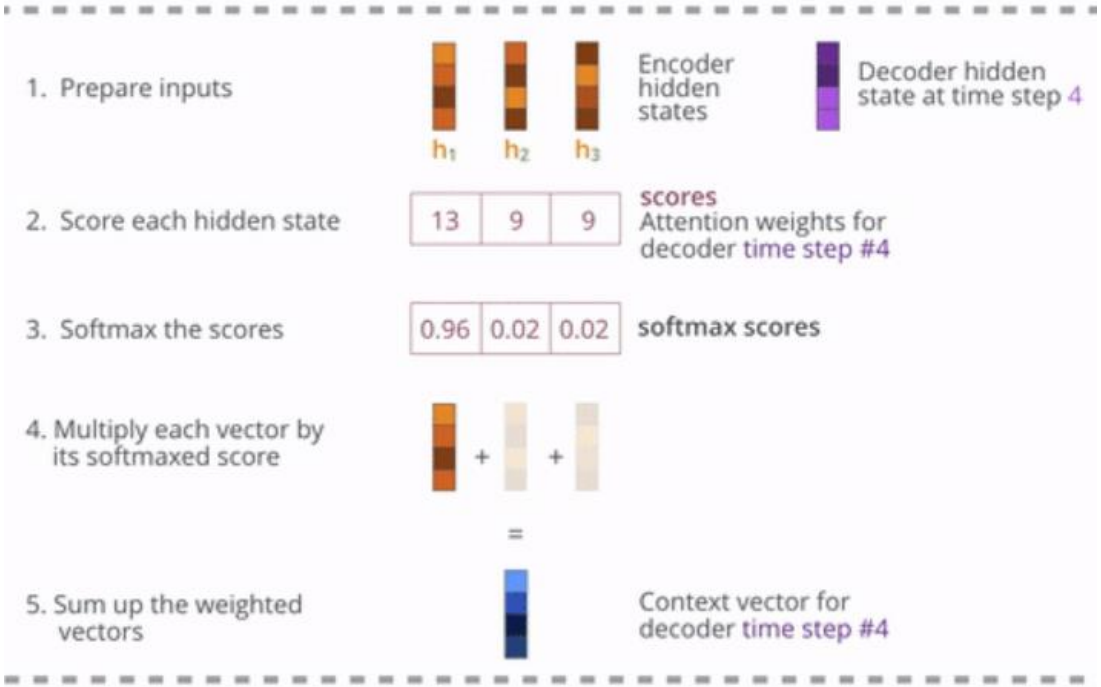


Transformers

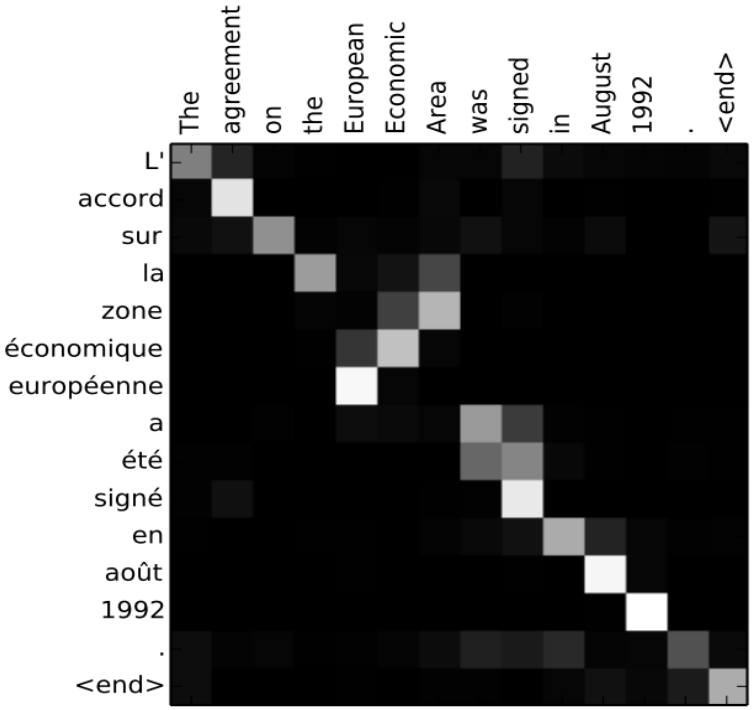
https://towardsdatascience.com/transformers-141e32e69591

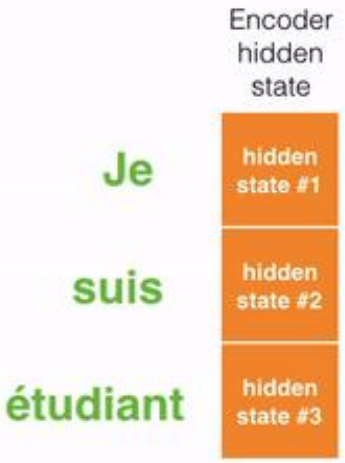
Idées principales

- Mécanisme d'attention ➔



- Attention maps :



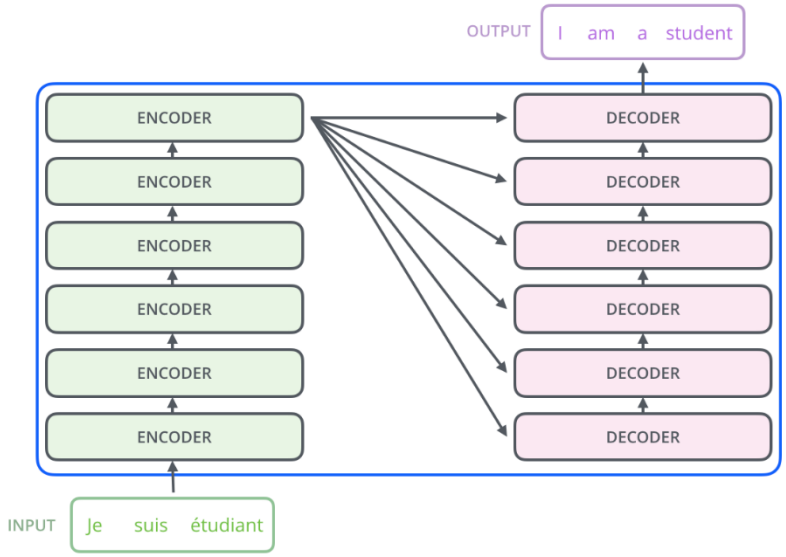


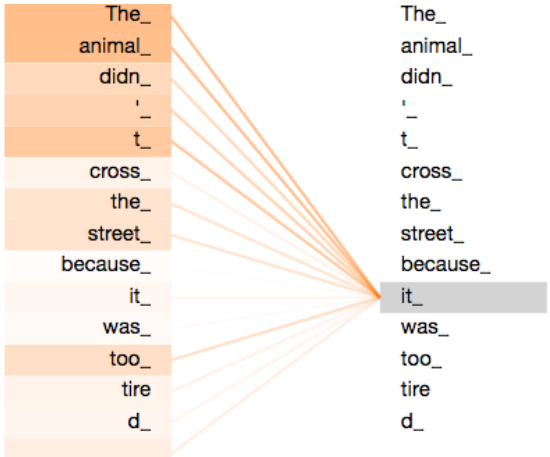
Transformers

(appliquée à la traduction automatique – tâche similaire à POS en fait !)

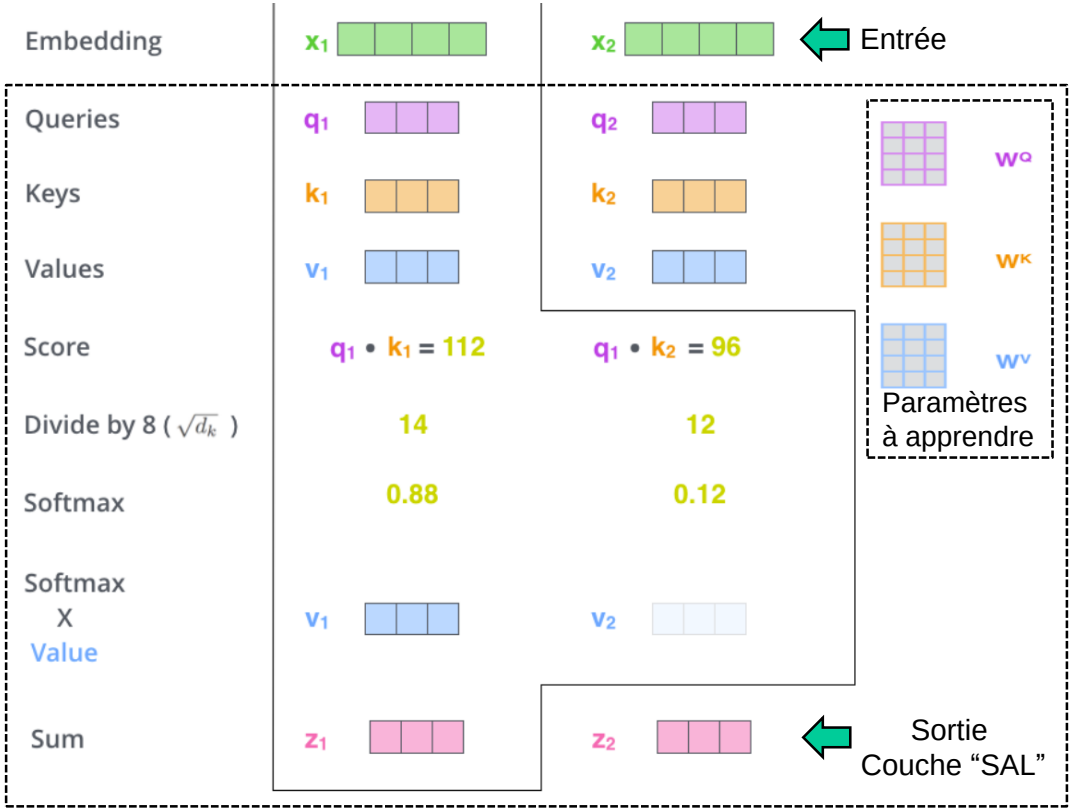
Empilement de Encodeurs / Decodeurs (remplacement des RNN)

- Tous les encodeurs et décodeurs ont la même architecture
- 2 couches principales : Self-attention + Feed-Forward





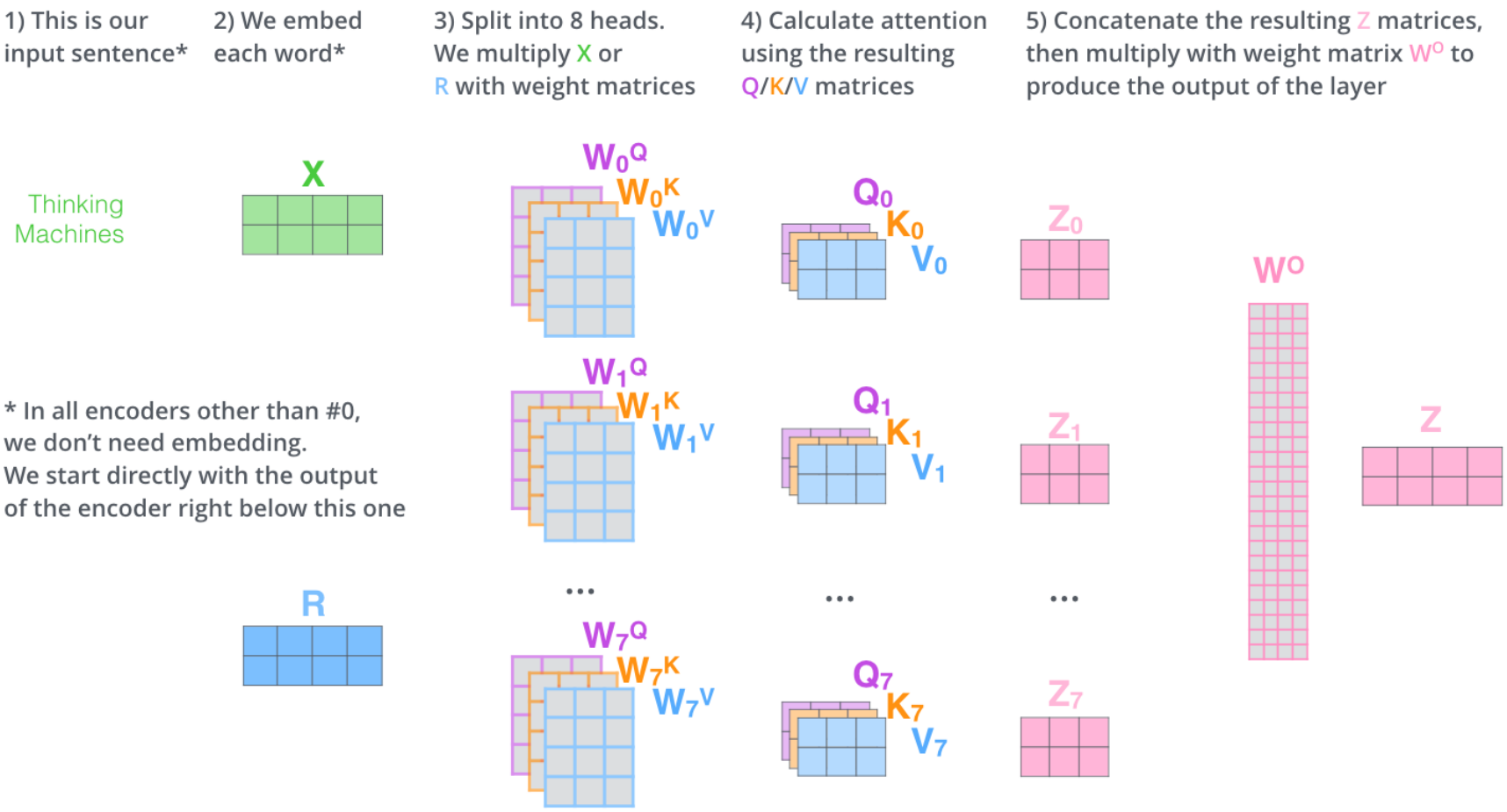
- **Mécanisme ‘self-attention Layer’**
  - Avec triplet : Query, Key, Value



Transformers – Coté Encodeur

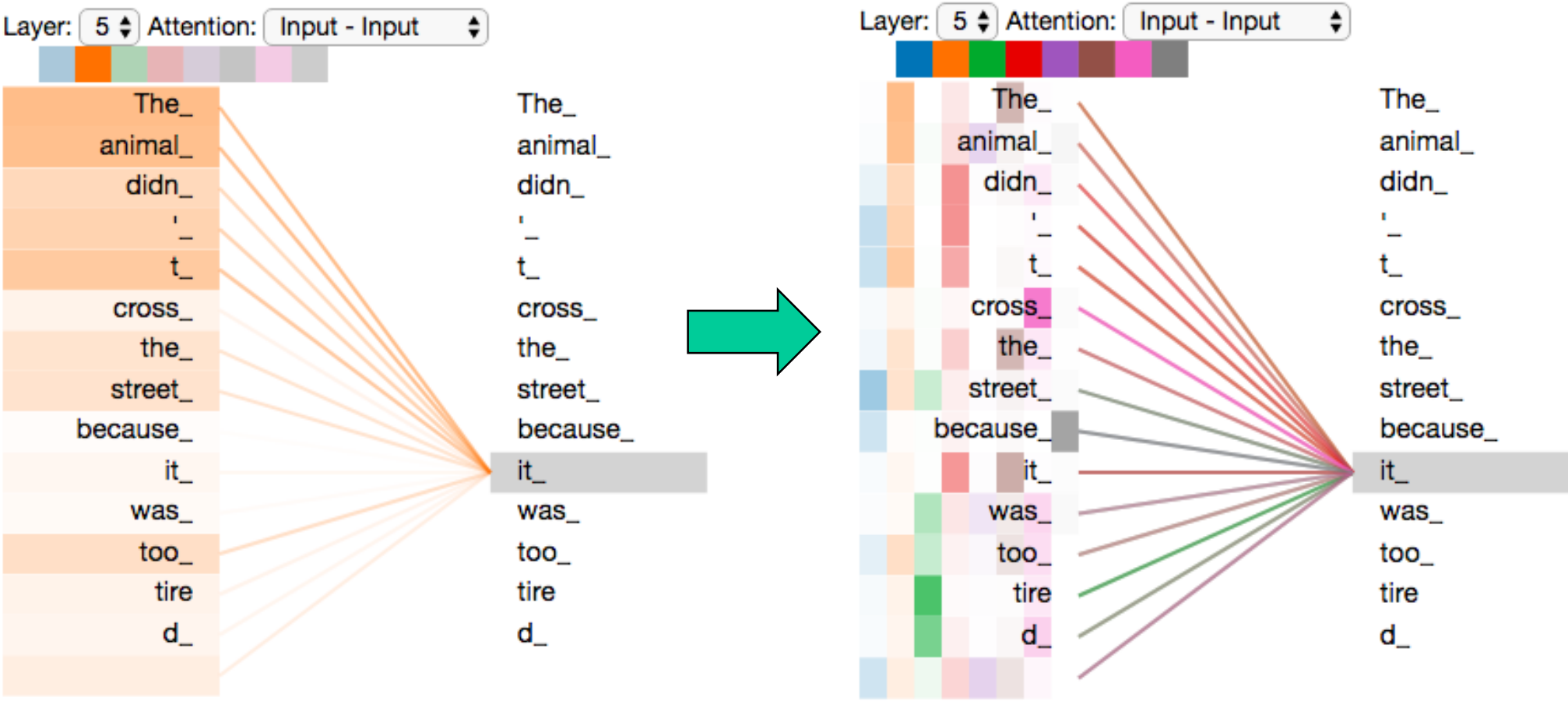
http://jalammar.github.io/illustrated-transformer/

- **Self attention** → Multi-Headed attention



Transformers – Coté Encodeur

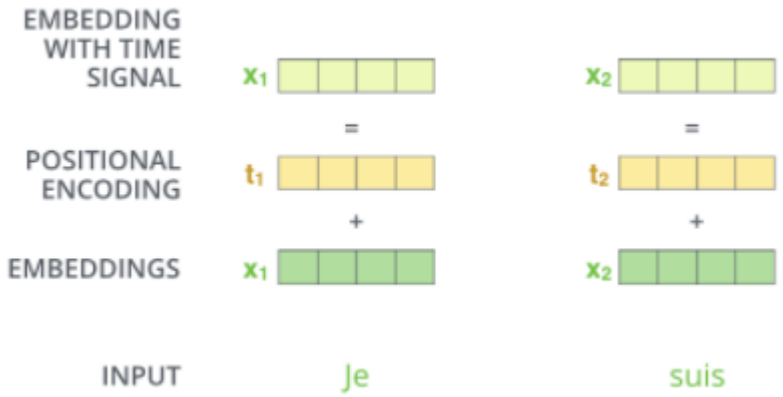
- Self attention → Multi-Headed attention



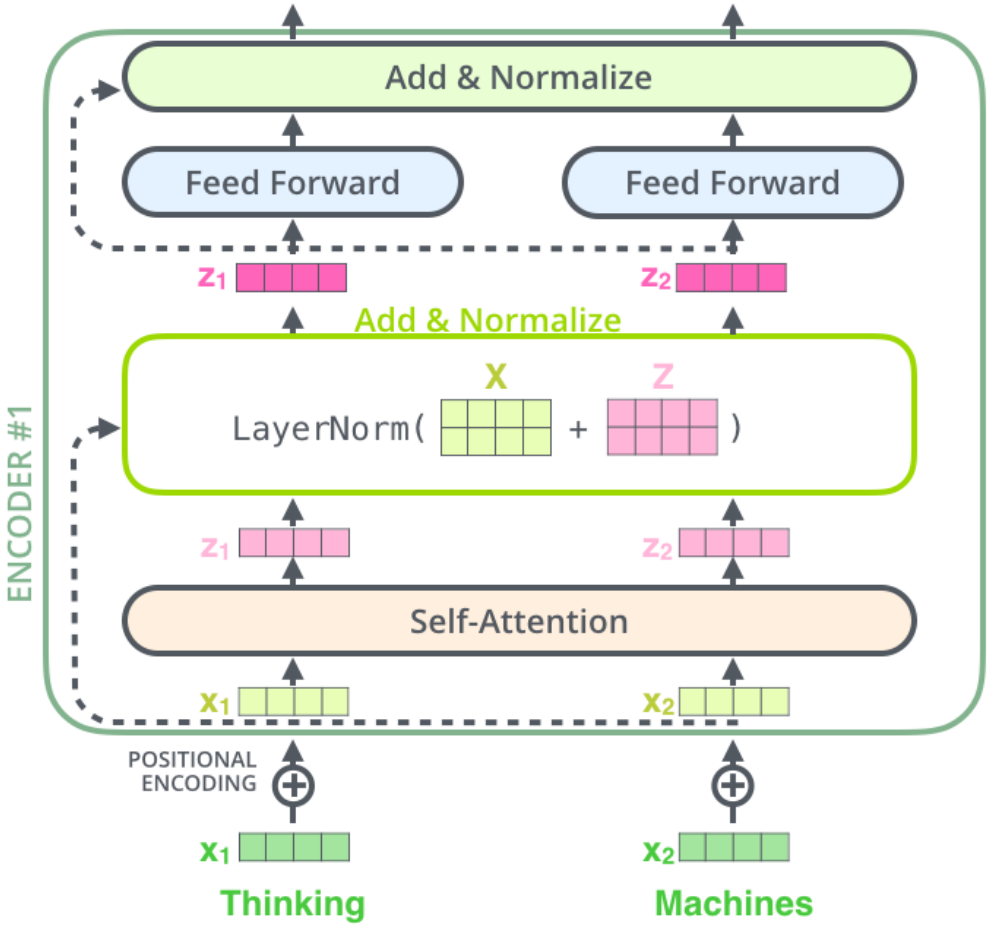
Transformers – Coté Encodeur

Les details

- Positional Encoding → Pour repré-
-senter l'ordre des mots

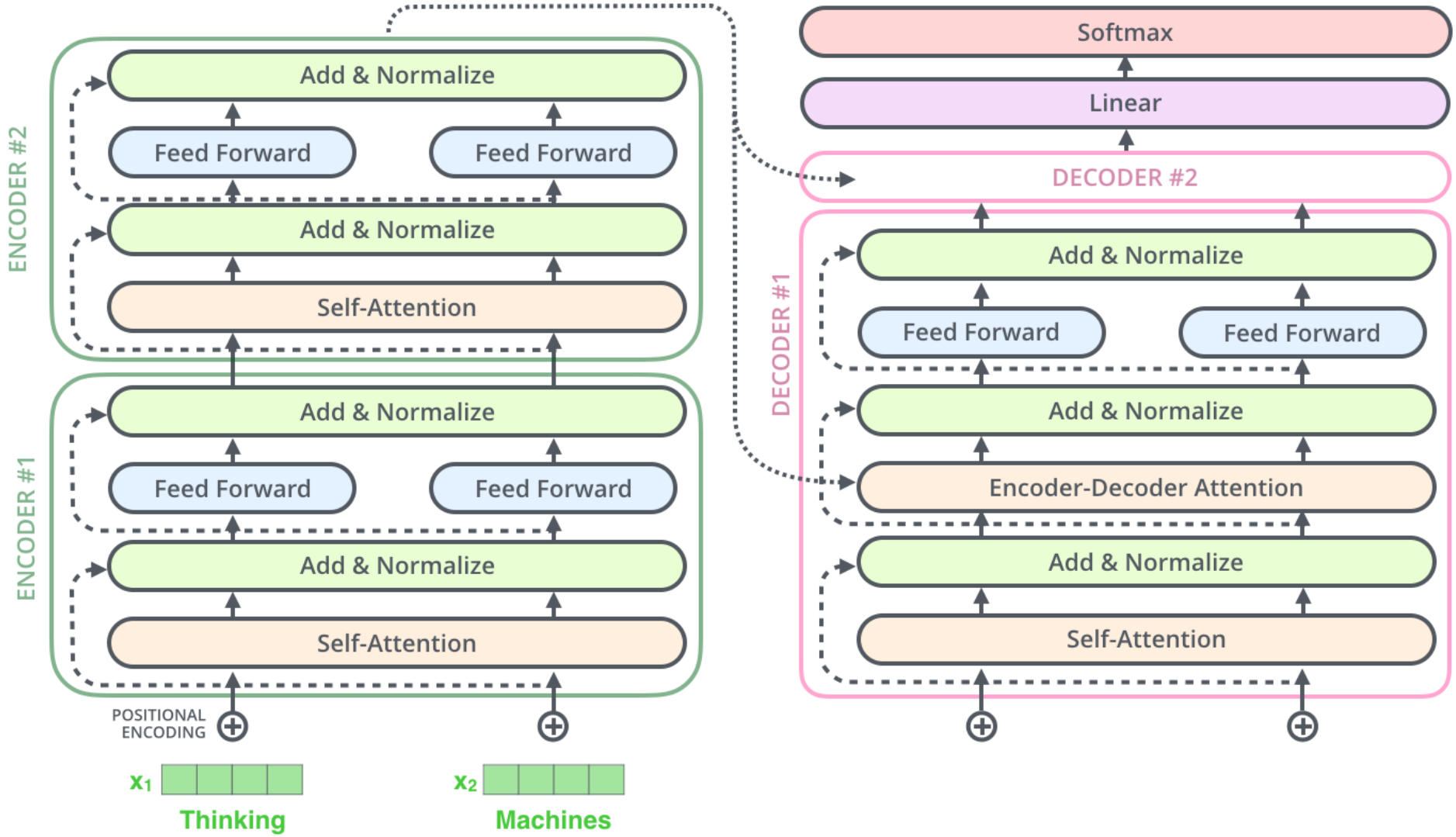


- Connexion résiduelle + Normalisation



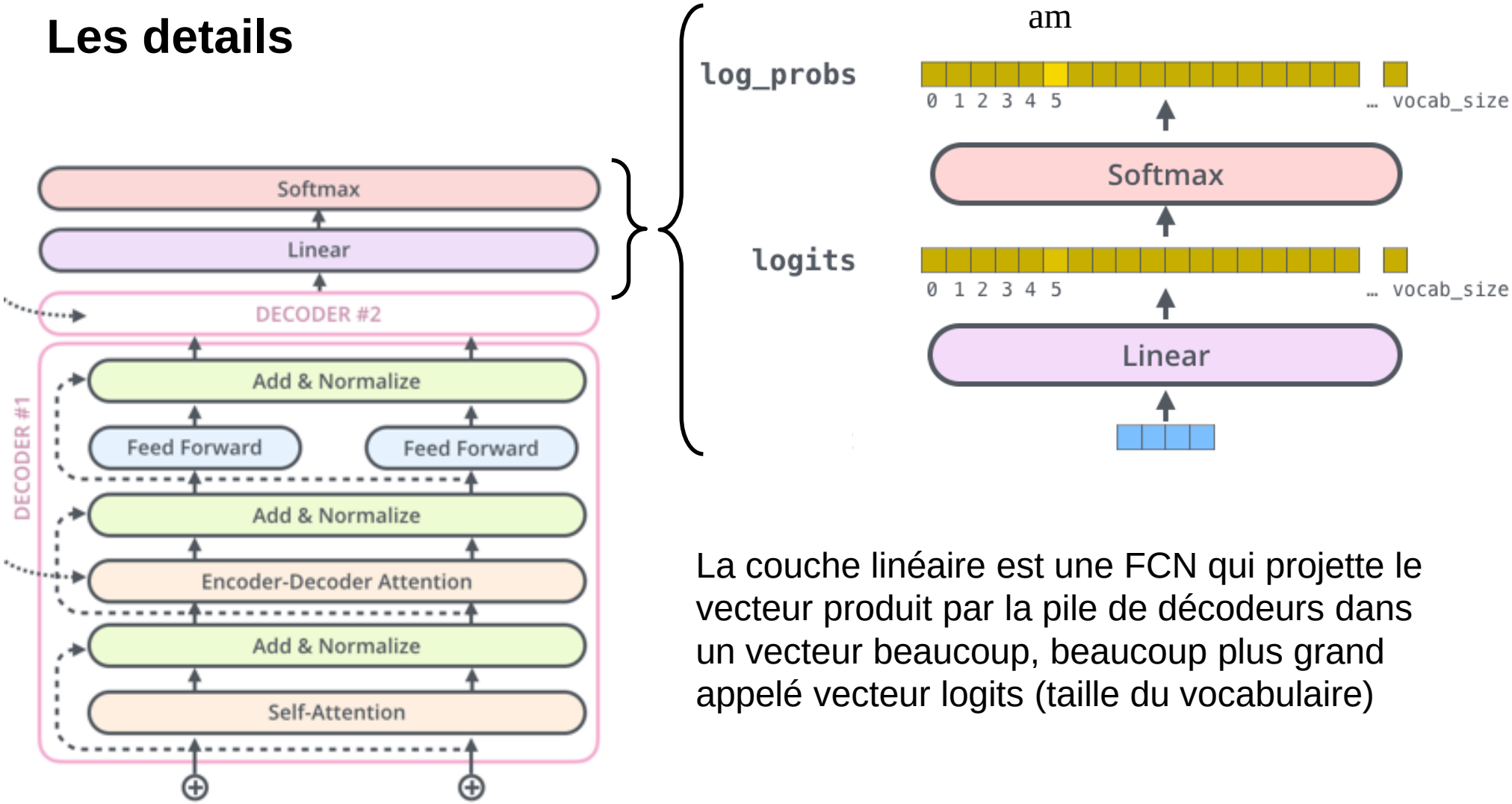
Transformers – Coté Décodeur

Les details



Transformers – Coté Décodeur

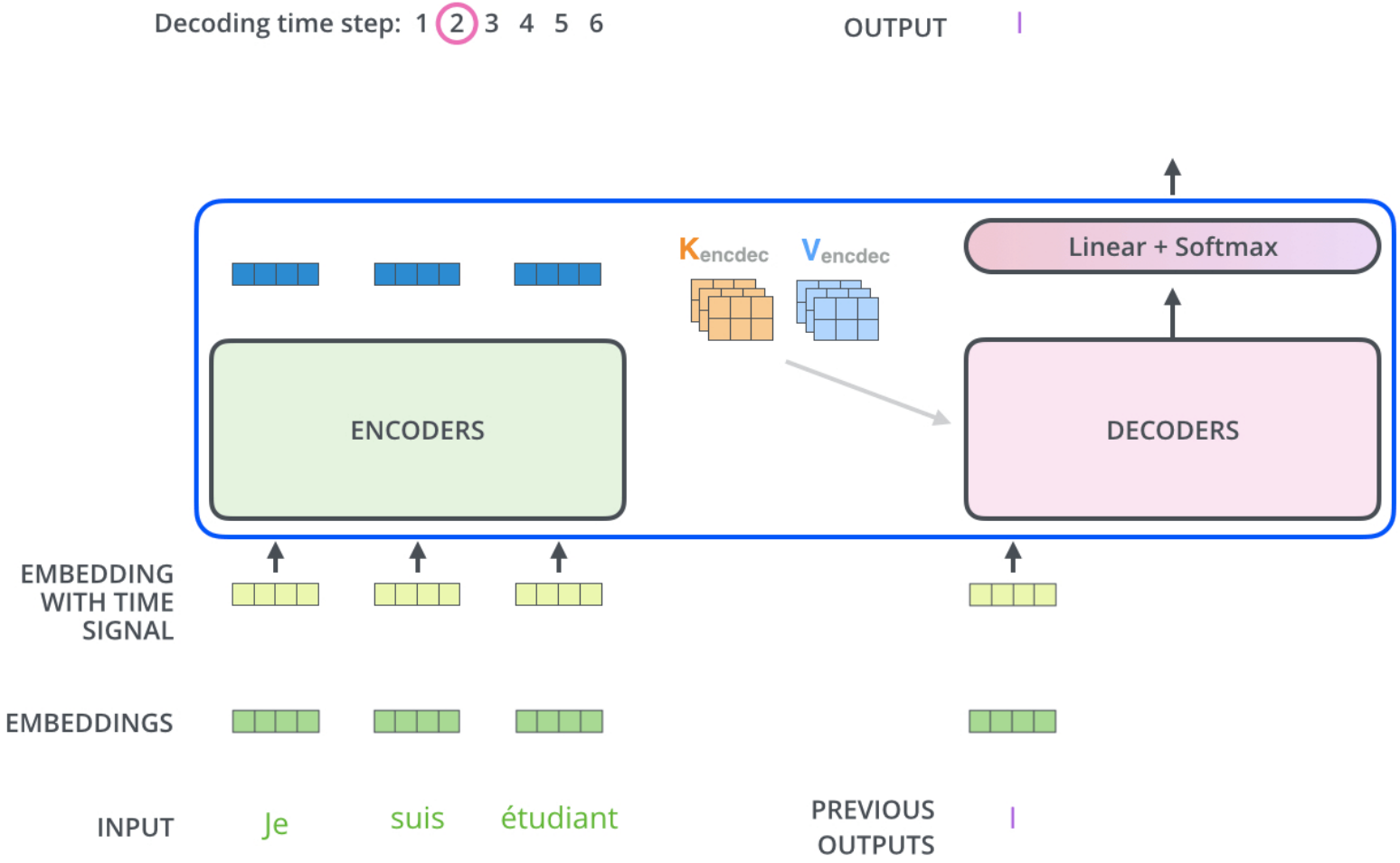
Les details



La couche linéaire est une FCN qui projette le vecteur produit par la pile de décodeurs dans un vecteur beaucoup, beaucoup plus grand appelé vecteur logits (taille du vocabulaire)

Transformers pour le NLP

En résumé



Bibliographie

- Indurkhya, N. and Damerau, F. J. (2010). Handbook of Natural Language Processing.Chapman & Hall/CRC, 2nd edition.
- Jurafsky, D. and Martin, J. H. (2019).Speech and Language Processing. https://web.stanford.edu/~jurafsky/slp3/
- Petrov, S., Das, D., and McDonald, R. (2012).A universal part-of-speech tagset.InProceedings of the Eighth International Conference onLanguage Resources and Evaluation (LREC'12), pages 2089–2096, Istanbul, Turkey. European Language Resources Association (ELRA).
- Taylor, A., Marcus, M., and Santorini, B. (2003).The Penn Treebank: An Overview, pages 5–22.Springer Netherlands, Dordrecht
- http://jalammar.github.io/illustrated-transformer/
- https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/
- https://towardsdatascience.com/transformers-141e32e69591

Outils :

- https://nlp.stanford.edu/software/tagger.shtml
- https://github.com/sloria/textblob
- https://www.nltk.org/api/nltk.tag.html
- https://spacy.io/https://github.com/clips/pattern
- …