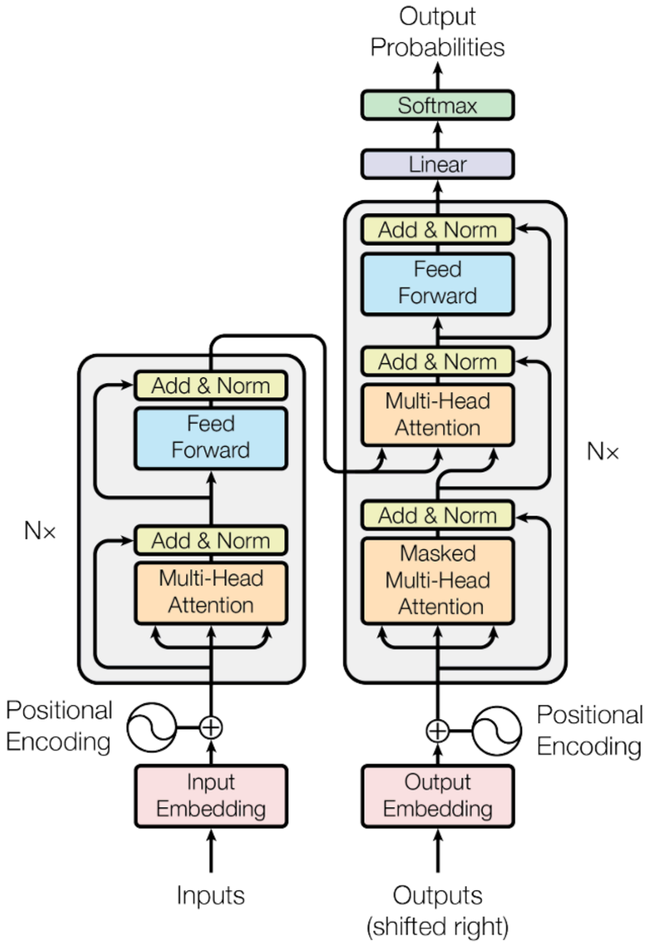
Apprentissage Automatique I
MATH60629

Attention et Transformeurs
— Semaine #10



Attention Is All You Need, Vaswani et al.
https://arxiv.org/pdf/1706.03762

Transformeurs

- Une architecture d'apprentissage profond
  - Développé en 2017 (par des chercheurs de Google)
    - Rapidement utilisé pour modéliser des données séquentielles (texte)—les LLM sont des transformeurs
  - Utilise le mécanisme d'*attention*

* La majorité des idées/diapos/figures viennent de David Berger (vous les reverrez peut-être en ML #2).

Plan de la séance

- Revue des RNN et des RNN bidirectionnels
- Concept de l'*attention*
- Bloc Transformeur
- Exemples de transformeurs en pratique

- Pour la séance, on imagine que nous modélisons de données textuelles
- Je vais utiliser les termes mots et tokens de manière synonyme
- En pratique, les mots sont souvent divisés en plusieurs tokens (chaque token est donc formé de quelques lettres)
    - Nous n’aborderons pas la « tokenization », c’est-à-dire comment l’on divise les mots en token

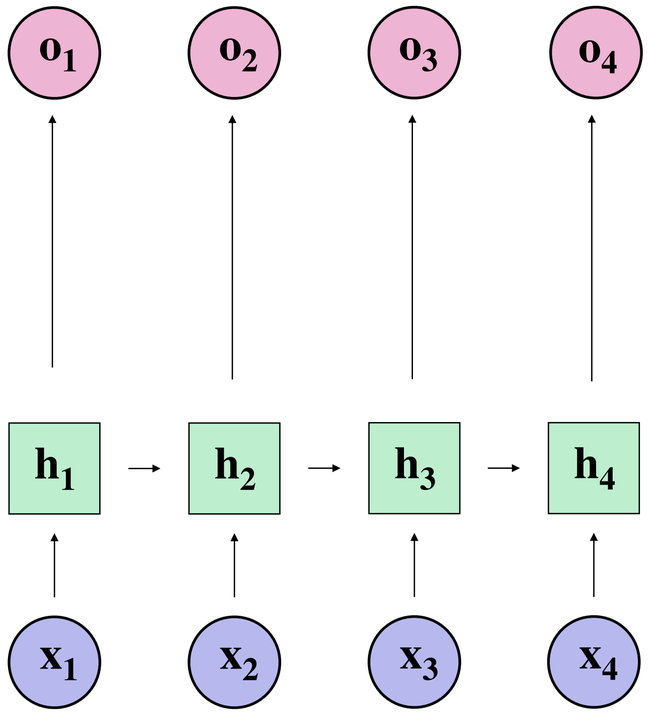
RNNs

Pour un pas de temps

x_i Data

h_i Couche cachée (Gauche à droite)

o_i Prédiction



- Les données sont traitées de manière séquentielle
- À chaque pas de temps, l'état caché h_t doit contenir toute l'information venant des mots précédents $x_1 : x_{t-1}$

RNN bidirectionnel

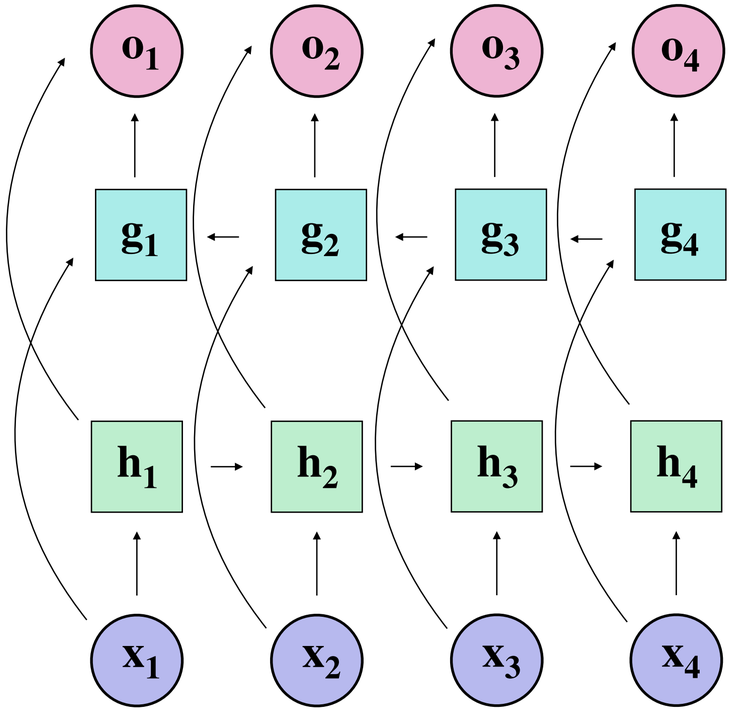
Pour un pas de temps

x_i Données

h_i Couche cachée (Gauche à droite)

g_i Couche cachée (Droite à gauche)

o_i Prédiction



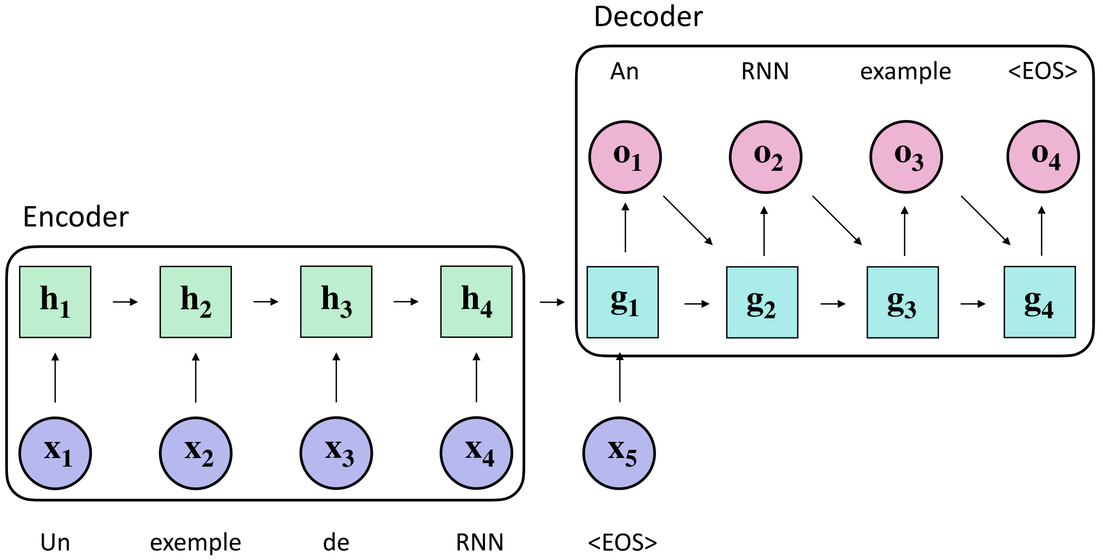
- h_tLes données sont traitées de manière séquentielle
- À chaque pas de temps, l'état caché doit contenir toute l'information venant des mots précédents $x_1 : x_{t-1}$
- À chaque pas de temps, l'état caché g_t doit contenir toute l'information venant des mots suivants $x_t : x_T$

Difficulté:

- Explosion et disparition du gradient
  - Les états cachés vont plutôt apprendre de l'information de leurs voisins proches

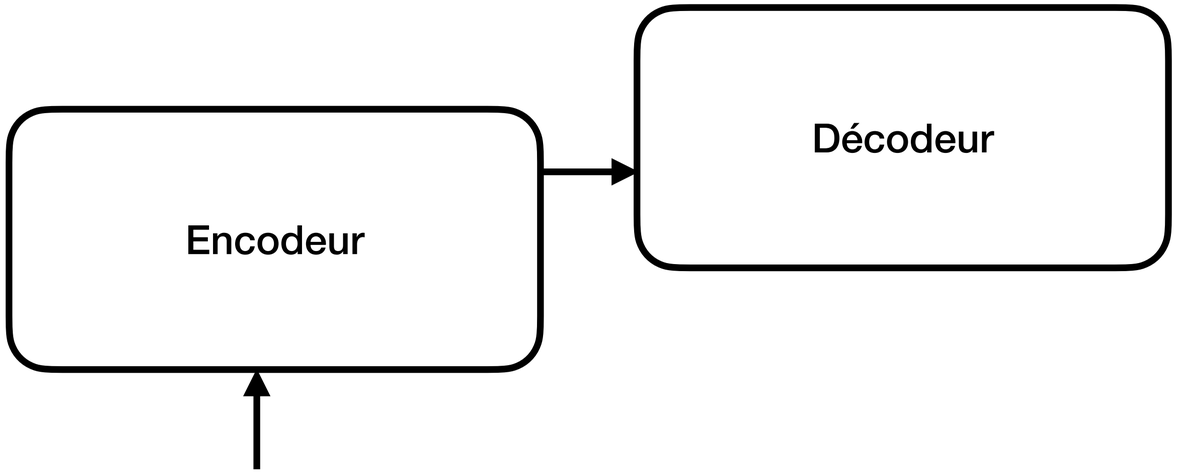
Encodeur-Décodeur
(Aussi connu sous le terme Sequence à Sequence — Seq2Seq)

- **Comment faire si la longueur des séquences en entrée n'est pas la même qu'en sortie? Par exemple, pour la traduction automatique.**



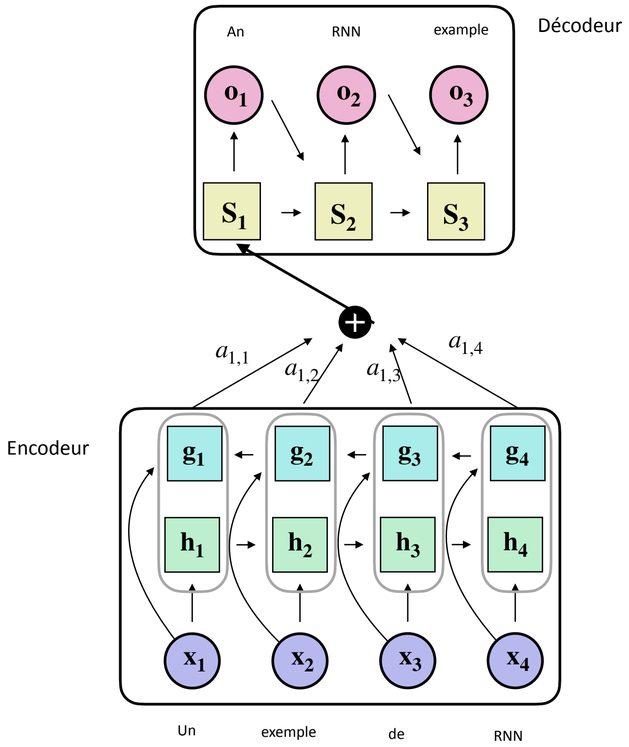
- h_4 **doit contenir toute l'information de la séquence à traduire**
  - **C'est un gout d'étranglement (*bottleneck)*. Sa taille (nombre de neurones) est donc importante.**

Problème

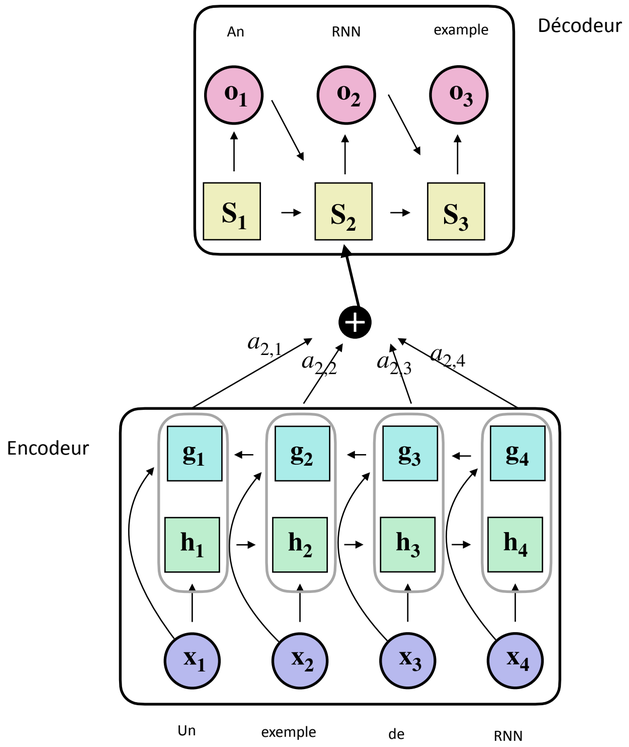


- **Difficile d'encoder beaucoup d'information**
- **On pourrait tenter de segmenter la séquence d'entrée (p. ex. une phrase à la fois)**
  - **Certaines tâches (notamment la traduction) ont besoin de cohérence locale et globale**
- **À la place: on décode mot par mot et à chaque on s'appuie sur les mots utiles de la séquence d'entrée**

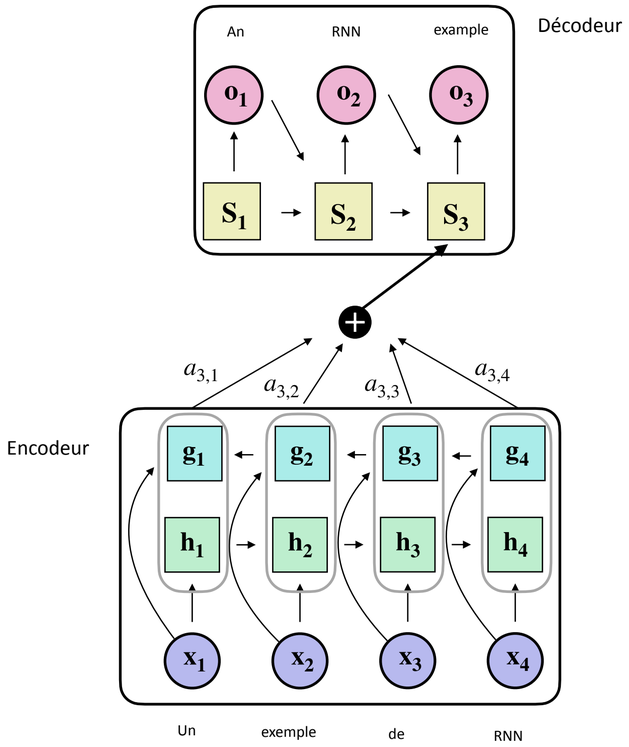
(Soft) Attention

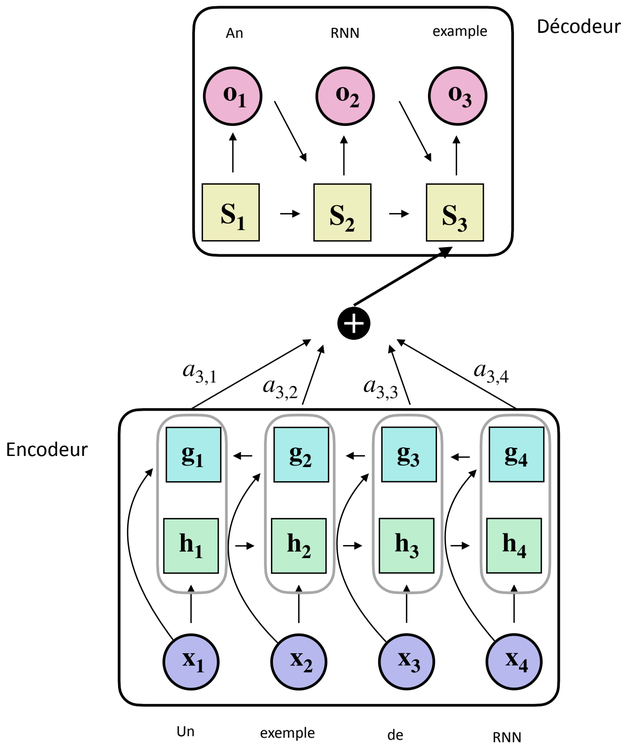


(Soft) Attention



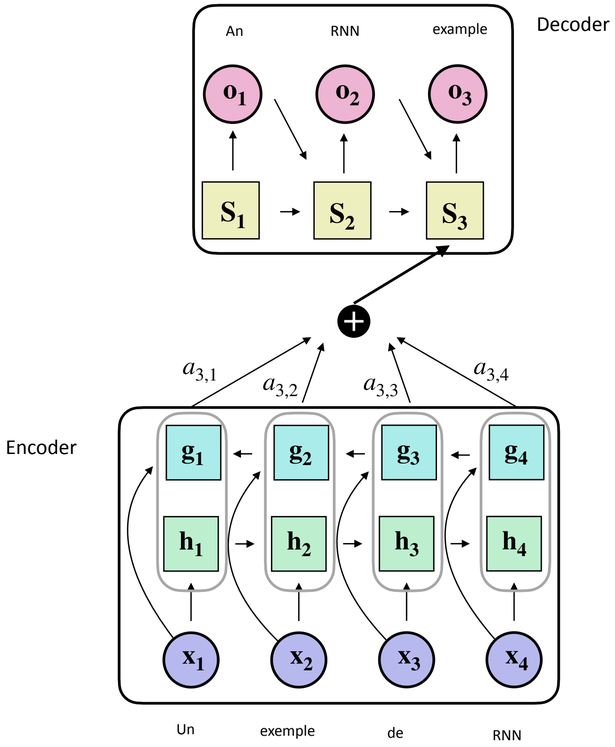
(Soft) Attention





Avantages:

- *Plus de goulot d'étranglement*
- Le décodeur choisit à chaque pas de temps la combinaison des mots de séquence d'entrée
- La représentation latente est proportionnelle à la longueur de la séquence
- Peut modéliser des dépendances d'un plus grand voisinage



Détails mathématiques:

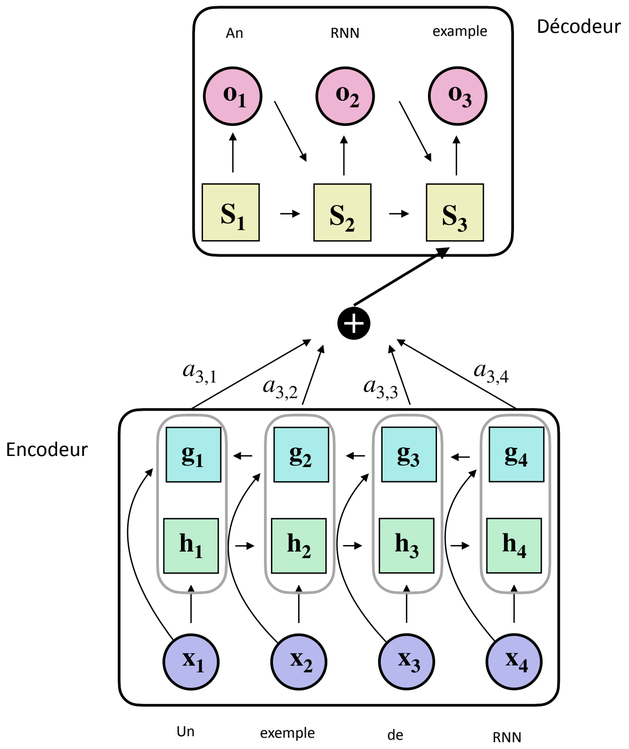
Sans attention: $f(s_t)$

$$\mathbb{P}(\mathbf{o_t} \mid \mathbf{o_{1:t-1}}, \mathbf{x_{1:T}}) = f(\mathbf{s_t}, \mathbf{c_t}),$$

Où:

$$\mathbf{c_t} = \sum_{\mathbf{j=1}}^{\mathbf{T_x}} \mathbf{a_{t,j}} \cdot [\mathbf{g_j}, \mathbf{h_j}]$$

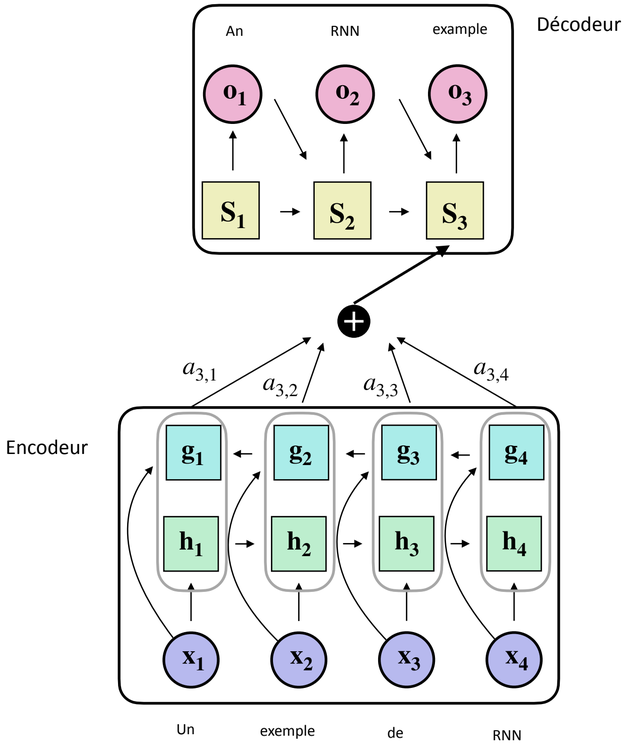
Attention
- Au pas de temps t (sortie)
- Représentation cachée j (entrée)



Détails mathématiques:

Les valeurs de l'attention sont obtenus:

$$\mathbf{a_{t,j}} = \Big(\text{softmax}(\mathbf{e_{tj}})\Big)_{\mathbf{j}}$$



Détails mathématiques:

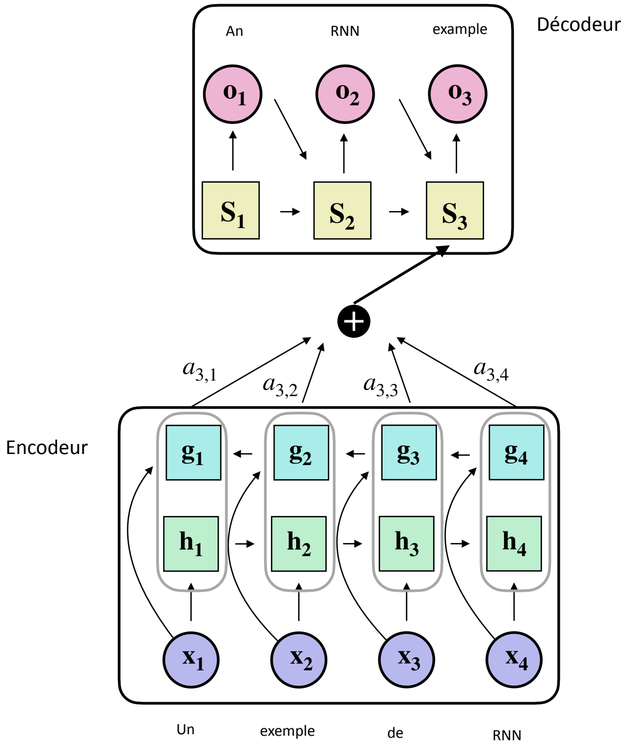
Les valeurs de l'attention sont obtenus:

$$\mathbf{a_{t,j}} = \Big(\text{softmax}(\mathbf{e_{tj}})\Big)_{\mathbf{j}}$$

où

$$\mathbf{e_{tj}} = \alpha(\mathbf{s_{t-1}}, \mathbf{h_j}, \mathbf{g_j})\,.$$

Function modélise la similarité entre Une représentation en sortie et une en entrée: p. Ex., $e_{ij} = s_{i-1} \cdot [hi; gi]$



Détails mathématiques:

Les valeurs de l'attention sont obtenus:

$$\mathbf{a_{t,j}} = \Big(\text{softmax}(\mathbf{e_{tj}})\Big)_{\mathbf{j}}$$

où

$$\mathbf{e_{tj}} = \alpha(\mathbf{s_{t-1}}, \mathbf{h_j}, \mathbf{g_j})\,.$$

Function modélise la similarité entre Une représentation en sortie et une en entrée: p. Ex., $e_{ij} = s_{i-1} \cdot [hi; gi]$

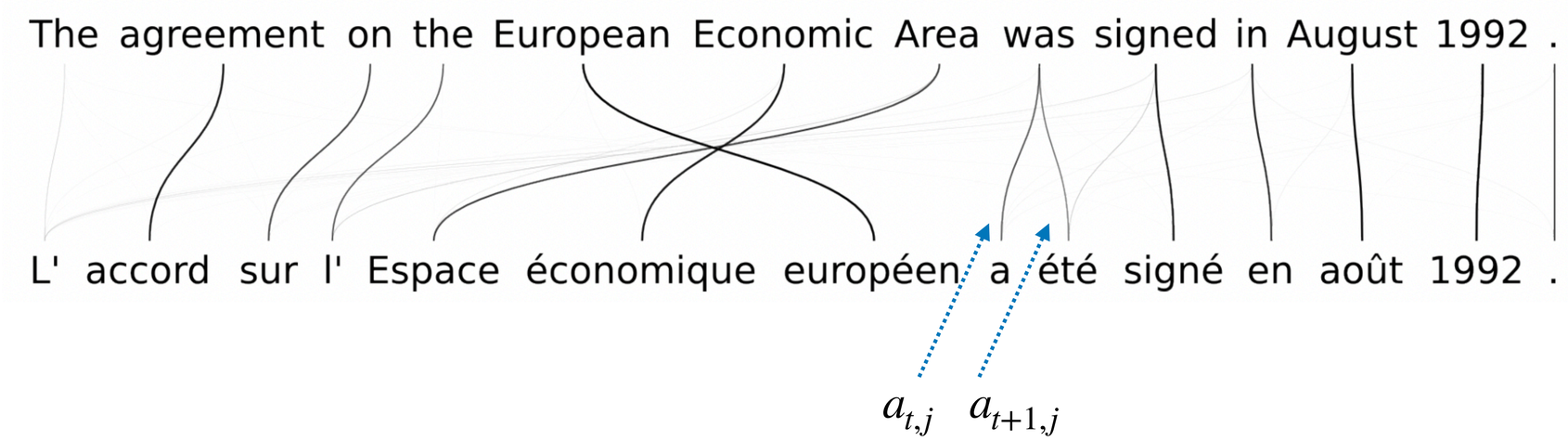
La fonction $\alpha(\,\cdot\,)$ peut, par exemple, être un MLP:

$$\alpha(\mathbf{s_{i-1}}, \mathbf{h_i}, \mathbf{g_i}) = \mathbf{v_a^\top} \tanh\Big(\mathbf{W}_\alpha \mathbf{s_{i-1}} + \mathbf{U}_\alpha[\mathbf{h_i}, \mathbf{g_i}]\Big)\,.$$

$\mathbf{W}_\alpha, \mathbf{U}_\alpha, \mathbf{v}_\alpha$: paramètres

Visualisation de l’attention

âche de traduction (EN -> FR)

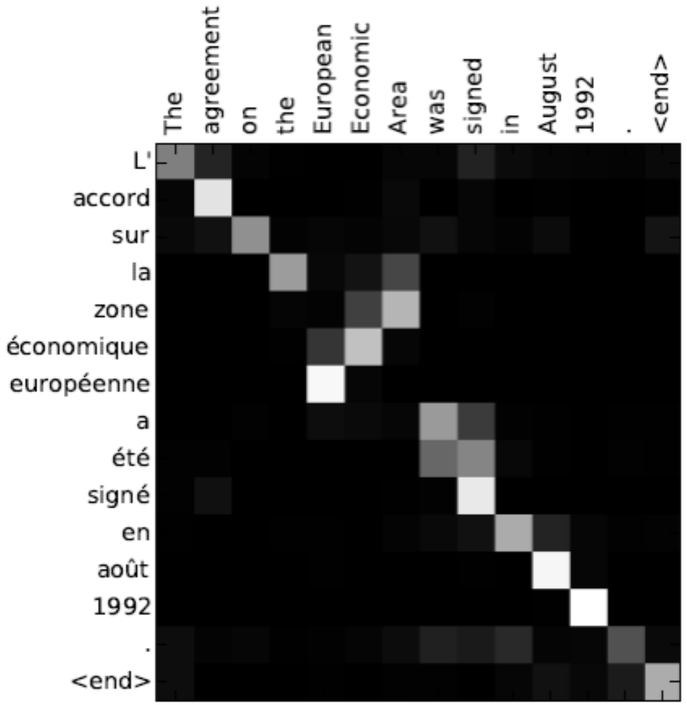


Visualizing attention

Tâche de traduction (EN -> FR)
- Matrice des valeurs de l’attention pour chaque paire de mots entrée-sortie

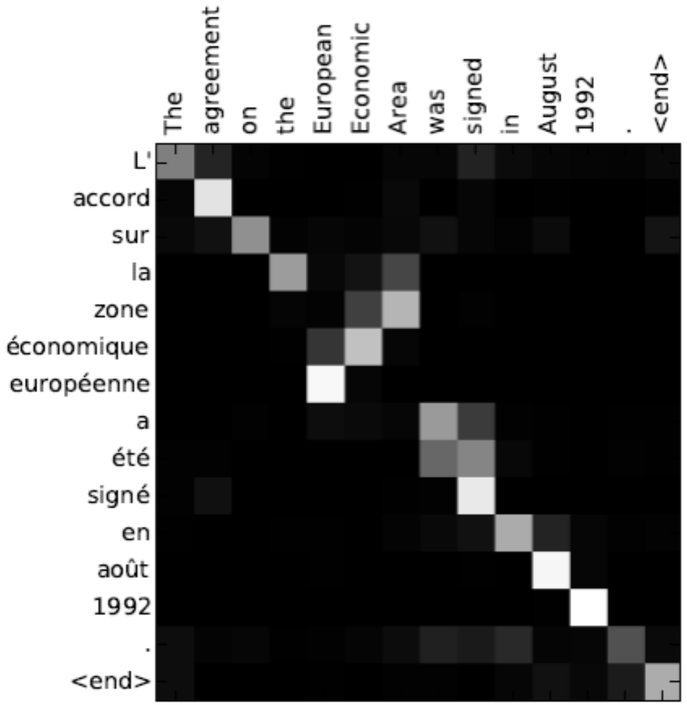
Visualizing attention

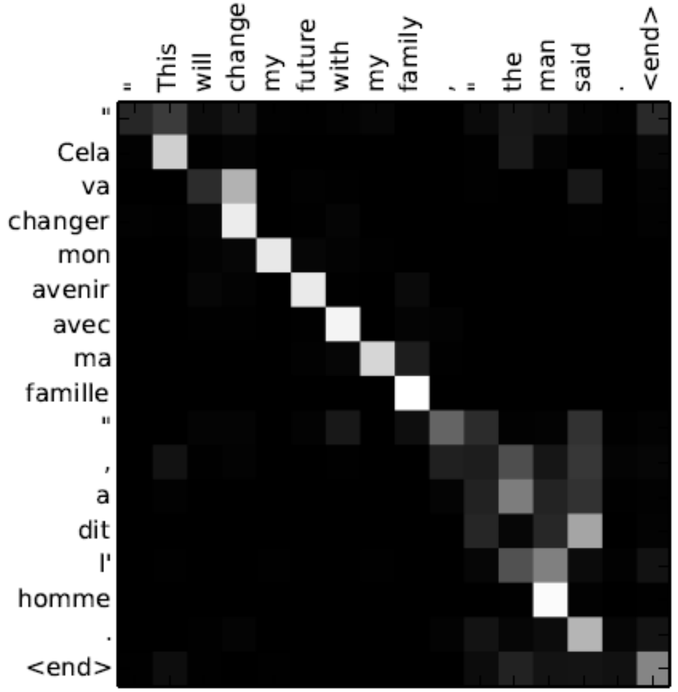
Tâche de traduction (EN -> FR)
- Matrice des valeurs de l’attention pour chaque paire de mots entrée-sortie



Visualizing attention

Tâche de traduction (EN -> FR)
- Matrice des valeurs de l’attention pour chaque paire de mots entrée-sortie

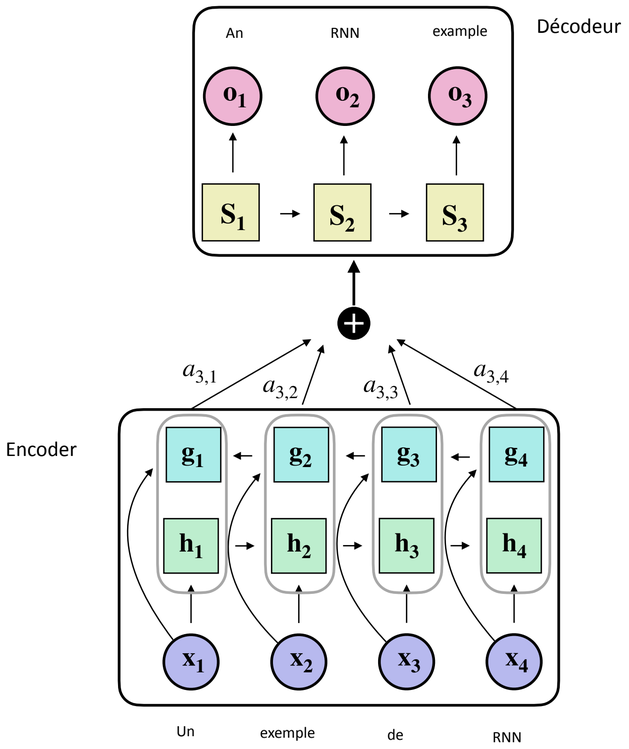




Sommaire du Soft Attention

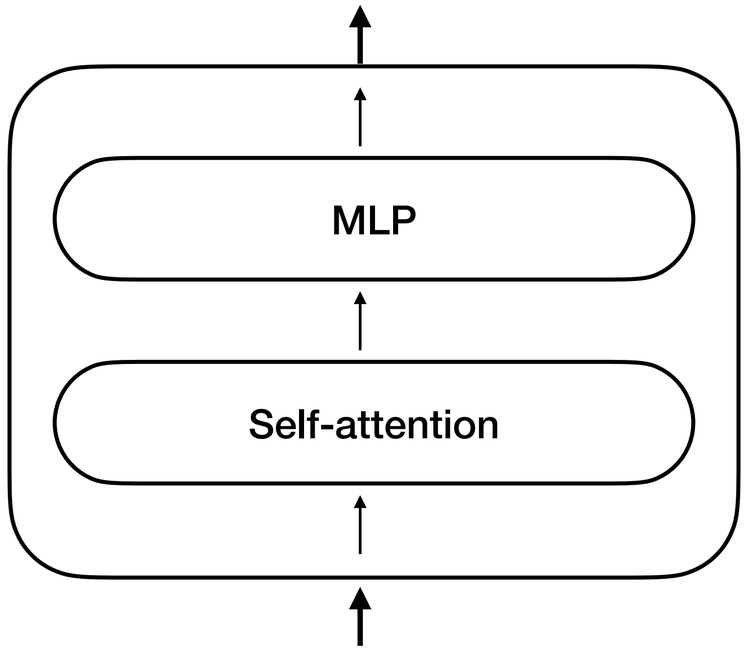
- Obtient dynamiquement la pondération d’un contexte
- Empiriquement: Fonctionne pour des longueur de séquence content ~100 pas de temps (quelques dizaines de mots)

Vers les Transformeurs



- Utilise des opérations séquentielles (h_i, g_j, S_t) et parallèles (attention)
  - Reste coûteux
- Peut-on obtenir un modèle complètement parallèle pour modéliser des données séquentielles?

Bloc Transformeur



William Shakespeare (c. 23[a] April 1564 – 23 April 1616)[b] was an English playwright, poet and actor.
He is widely regarded as the greatest writer in the English language and the world's pre-eminent dramatist.[3][4][5]
He is often called England's national poet and the "Bard of Avon" (or simply "the Bard").
His extant works, including collaborations, consist of some 39 plays, 154 sonnets,
three long narrative poems and a few other verses, some of uncertain authorship.

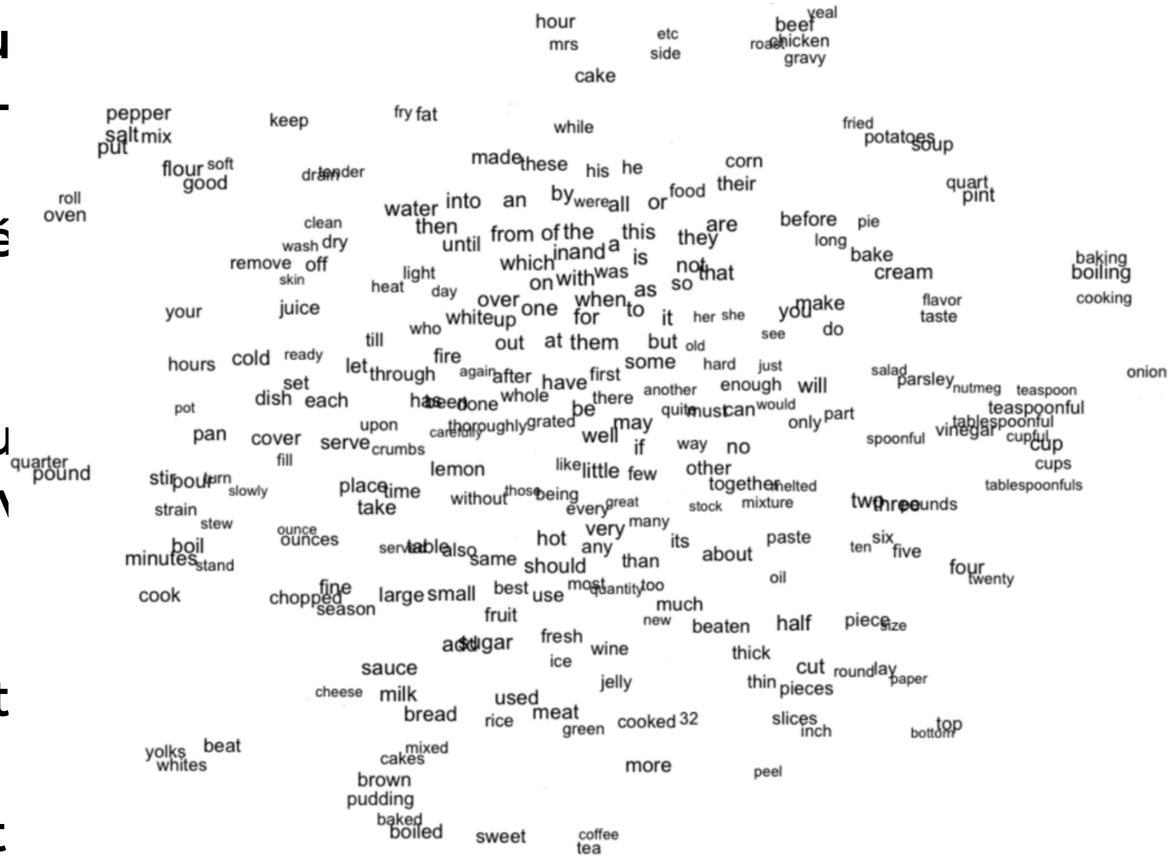
- **Un bloc transforme la représentation des mots de la séquence**

Word embedding

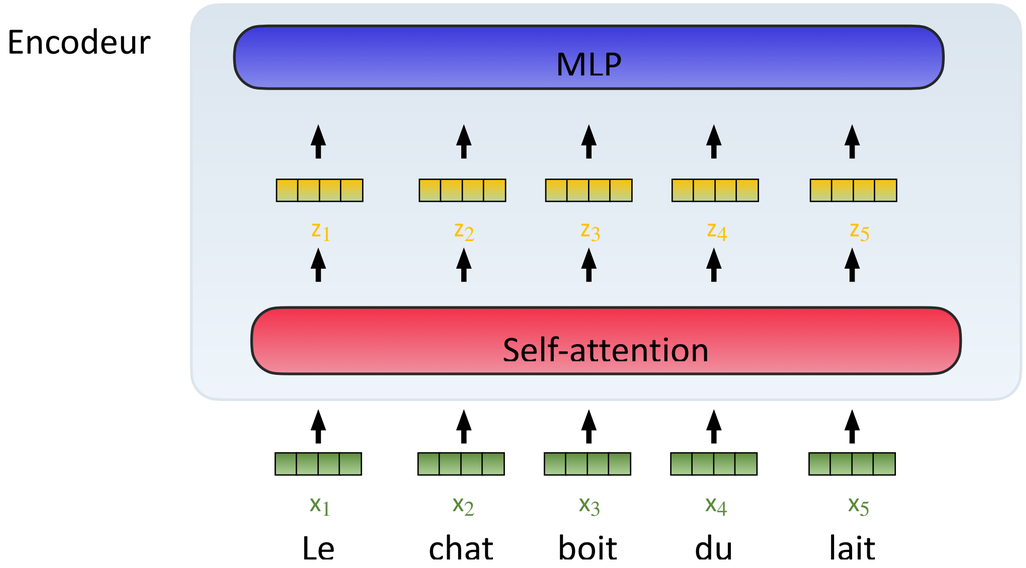
- Les mots correspondent à une variable catégorielle. On peut utiliser encodage 1-de-K (dummies)
- Cet encodage ne peut représenter les similarités entre les mots (p. ex., sel/poivre)
- Dans les RNNs/Transformeurs/etc. les mots sont représentés en utilisant un vecteur de valeurs continues (de taille D)
    - Cette représentation est apprise
    - Cette représentation peut être de plus petite dimensionnalité (D << K)

Word embedding

- Les mots correspondent à u
peut utiliser encodage 1-de-
- Cet encodage ne peut repré
mots (p. ex., sel/poivre)
- Dans les RNNs/Transformeu
représentés en utilisant un v
(de taille D)
  - Cette représentation est
- Cette représentation peut
dimensionnalité (D << K)



Self-attention



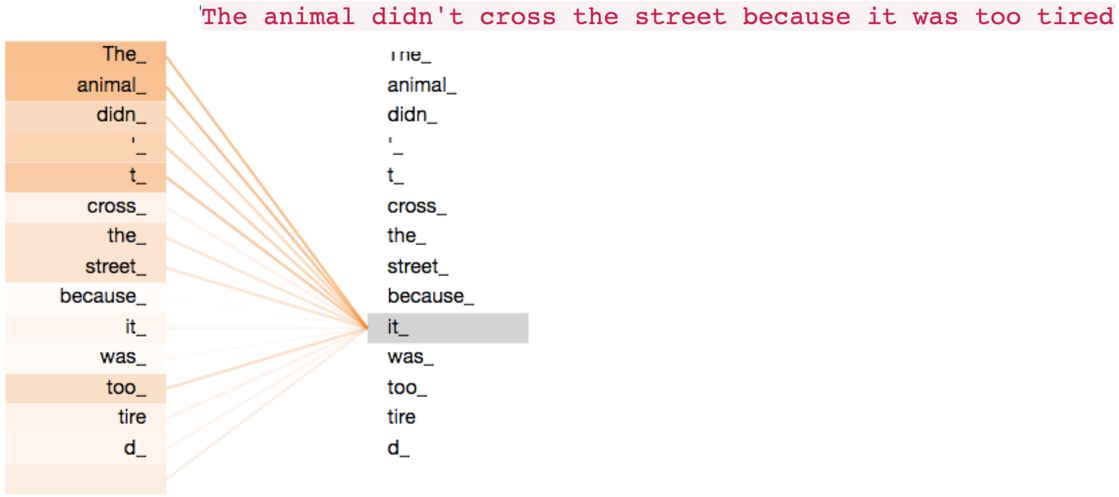
- **Le mécanisme de self-attention transforme donc la représentation de chaque mot (sont word embedding)**

Self-attention

- C’est le mécanisme d’attention où la séquence d’entrée est la même que la séquence de sortie.

  →La représentation de chaque mot est une combinaison de la représentation des autres mots dans la séquence

Exemple:



- Ce mécanisme est non supervisé (d’où son nom *self*)

Self-Attention

Détails mathématiques

(Rappel) Soft attention

$$\mathbf{a_{t,j}} = \left(\text{softmax}(\mathbf{e_{ij}})\right)_{\mathbf{j}},$$

Représentation du contexte du i'ème mot

$$e_{ij} = v_a^\top \tanh\left(\mathbf{W}_\alpha \mathbf{s_{i-1}} + \mathbf{U}_\alpha[\mathbf{h_i}, \mathbf{g_i}]\right).$$

Projection linéaire

Projection linéaire

Self-Attention

Détails mathématiques

(Rappel) Soft attention

$$\mathbf{a_{t,j}} = \Big(\text{softmax}(\mathbf{e_{ij}})\Big)_{\mathbf{j}},$$

Représentation du contexte du i'ème mot

$$e_{ij} = v_a^\top \tanh\left(\mathbf{W}_\alpha \mathbf{s_{i-1}} + \mathbf{U}_\alpha[\mathbf{h_i}, \mathbf{g_i}]\right).$$

Projection linéaire

Projection linéaire

Self attention

$$\mathbf{a_{i,j}} = \Big(\text{softmax}(\mathbf{e_{ij}})\Big)_{\mathbf{j}},$$

Où

$$\mathbf{e_{ij}} = \Big((\mathbf{x_i}\mathbf{W^k})(\mathbf{x_j}\mathbf{W^q})\Big)$$

Self-Attention

Détails mathématiques

(Rappel) Soft attention

$$\mathbf{a_{t,j}} = \Big(\text{softmax}(\mathbf{e_{ij}})\Big)_{\mathbf{j}},$$

Représentation du contexte du i'ème mot

$$e_{ij} = v_a^\top \tanh\left(\mathbf{W}_\alpha \mathbf{s_{i-1}} + \mathbf{U}_\alpha[\mathbf{h_i}, \mathbf{g_i}]\right).$$

Projection linéaire

Projection linéaire

Self attention

$$\mathbf{a_{i,j}} = \Big(\text{softmax}(\mathbf{e_{ij}})\Big)_{\mathbf{j}},$$

Où

$$\mathbf{e_{ij}} = \Big((\mathbf{x_i}\mathbf{W^k})(\mathbf{x_j}\mathbf{W^q})\Big)$$

Calcul de la sortie

$$\mathbf{c_t} = \sum_{\mathbf{j=1}}^{\mathbf{T_x}} \mathbf{a_{t,j}} \cdot [\mathbf{g_j}, \mathbf{h_j}]$$

Self-Attention

Détails mathématiques

(Rappel) Soft attention

$$\mathbf{a_{t,j}} = \Big(\text{softmax}(\mathbf{e_{ij}})\Big)_{\mathbf{j}},$$

Représentation du contexte du i'ème mot

$$e_{ij} = v_a^\top \tanh\left(\mathbf{W}_\alpha \mathbf{s_{i-1}} + \mathbf{U}_\alpha[\mathbf{h_i}, \mathbf{g_i}]\right).$$

Projection linéaire

Projection linéaire

Calcul de la sortie

$$\mathbf{c_t} = \sum_{\mathbf{j=1}}^{\mathbf{T_x}} \mathbf{a_{t,j}} \cdot [\mathbf{g_j}, \mathbf{h_j}]$$

Self attention

$$\mathbf{a_{i,j}} = \Big(\text{softmax}(\mathbf{e_{ij}})\Big)_{\mathbf{j}},$$

Où

$$\mathbf{e_{ij}} = \Big((\mathbf{x_i W^k})(\mathbf{x_j W^q})\Big)$$

X: la représentation en entrée des mots (NxD)
z_j: la représentation en sortie du j'ème mot (1xD)

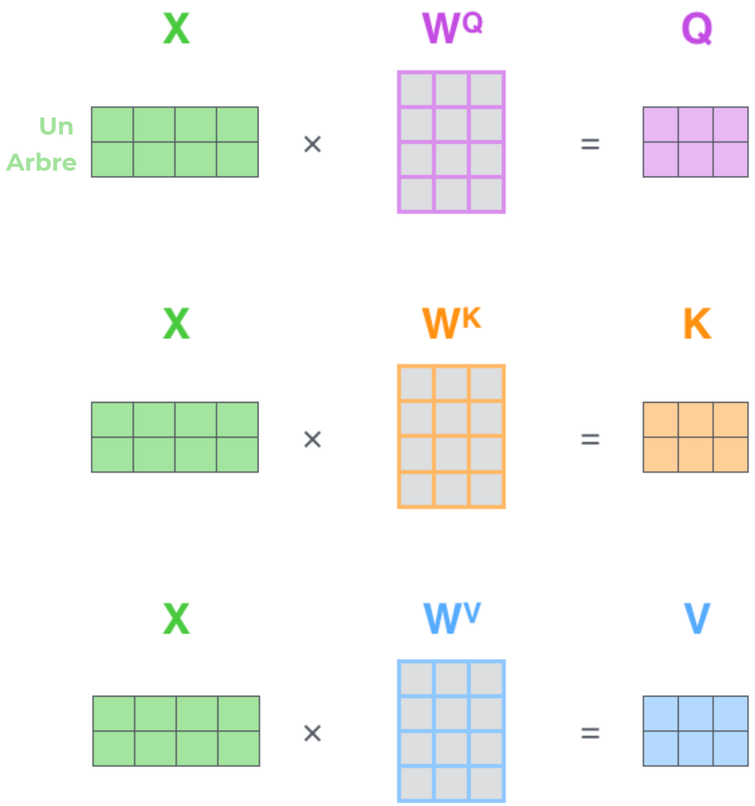
Calcul de la sortie

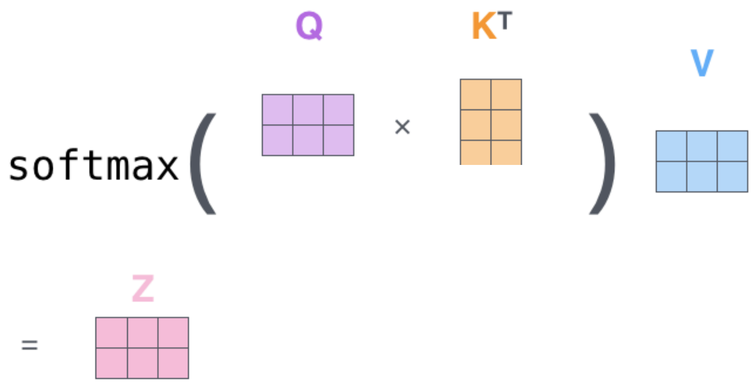
Souvent appelé "Key-Query-Value" attention

$$\mathbf{z_j} = \underbrace{(\mathbf{x_i W^k})}_{\text{Value}}\, \mathbf{softmax}(\underbrace{(\mathbf{x_i W^k})}_{\text{Key}}\, \underbrace{(\mathbf{x_j W^q})})$$

Self-Attention

Représentation visuelle
N=2, D=4





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

Multi-Head Self-Attention (MHSA)

- L’attention modélise une certaine similarité entre un mot et un ensemble
- On pourrait vouloir capturer différents types de similarités (p. ex., baleines et humains sont des mammifères, baleines et les requis vivent dans l’eau)
- On peut apprendre les paramètres de plusieurs mécanismes en même temps. Chaque mécanisme est appelé tête d’attention (attention head):

$$\mathbf{x'_j} = [\mathbf{z_j^1 z_j^2 \cdots z_j^H}]\mathbf{W^0}$$

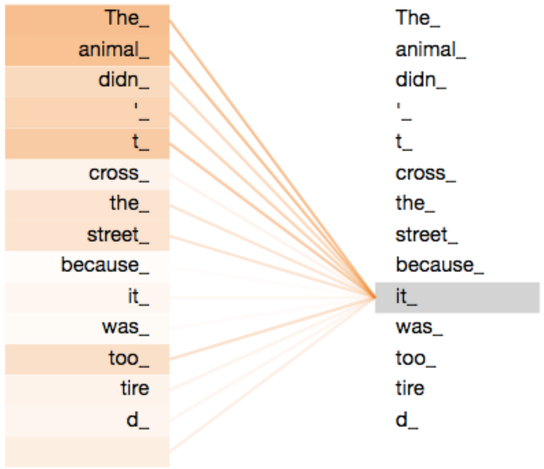
x'_j :la représentation du mot j à la sortie du self-attention (D)
z_j^h: la représentation du mot j venant de la tête d’attention h (D’)
H: le nombre de tête
W^0 : matrice de poids (D’H x D)

Multi-Head Self-Attention (MHSA)

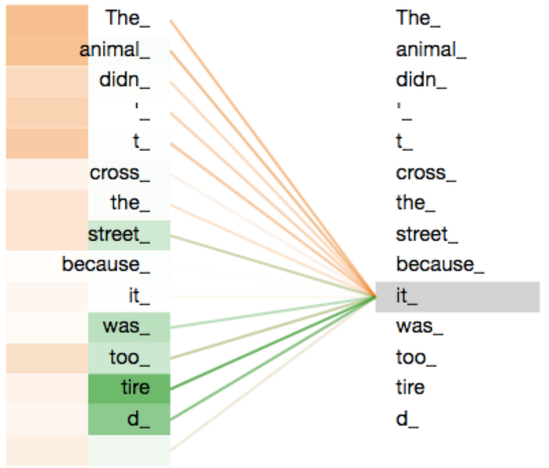
Exemple:

'The animal didn't cross the street because it was too tired'

One head



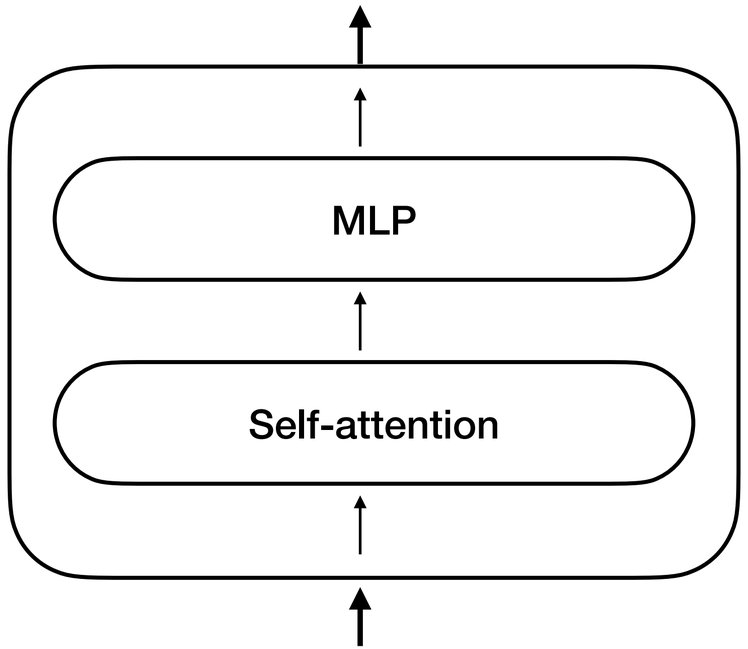
Two heads



Demo

- https://poloclub.github.io/transformer-explainer/
  - Montre l’attention (query, key, value)
  - Les mots peuvent être divisés en token

Bloc Transformeur



Opérations entre les caractéristiques des mots (D)

(Plutôt) Opérations entre les mots (N)

William Shakespeare (c. 23[a] April 1564 – 23 April 1616)[b] was an English playwright, poet and actor.
He is widely regarded as the greatest writer in the English language and the world's pre-eminent dramatist.[3][4][5]
He is often called England's national poet and the "Bard of Avon" (or simply "the Bard").
His extant works, including collaborations, consist of some 39 plays, 154 sonnets,
three long narrative poems and a few other verses, some of uncertain authorship.

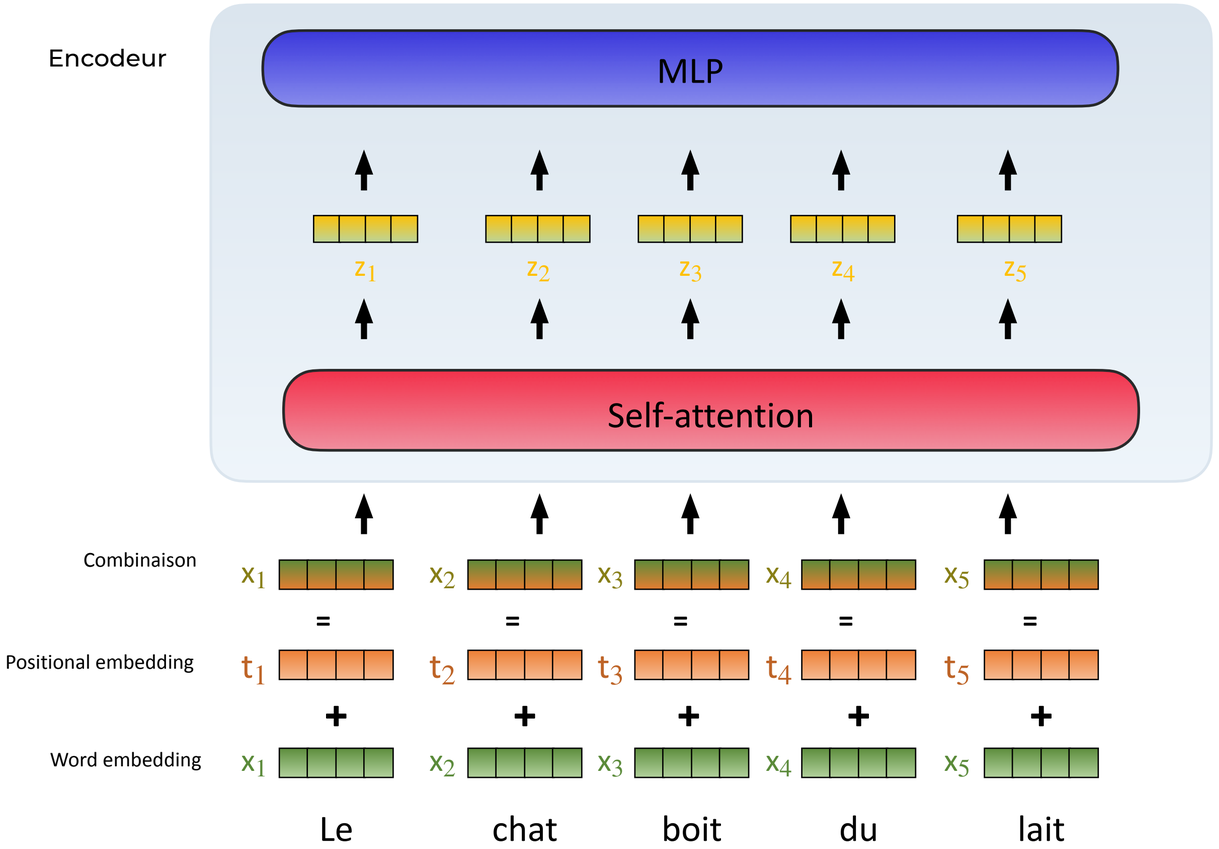
Transformeurs… transforment?

- Un bloc transformeur contient une couche de self-attention suivie d’un MLP
- Comment les utiliser dans des tâches (p. ex., traduction)?
- Chaque bloc peut être combiné, voire raffiné
  - Un bloc peut être utilisé comme un encodeur ou comme un décodeur
  - Pour décoder, quelques changements sont nécessaires

Quelques éléments d’information en plus

- Embedding de position
  - L’attention ne prend pas en compte la position des mots
    - La position d’un mot dans une séquence est souvent essentielle
      - “Le chat mange la plante.” est différent de “La plante mange le chat.”
  - Comme remède, on ajoute à chaque représentation initiale des mots, une représentation liée à la position du mot
    - Vecteur de valeurs continues (absolue ou relative)

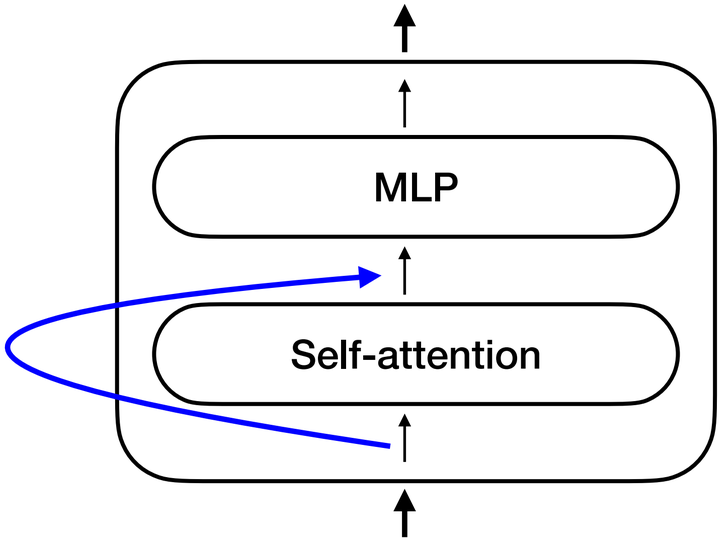
Word + Positional embeddings



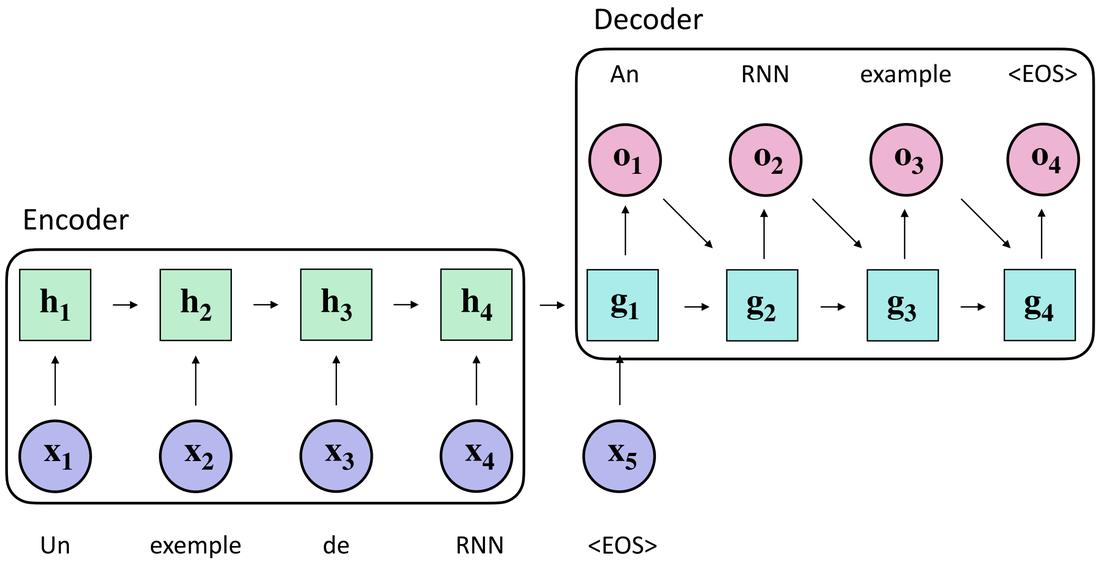
30

Quelques éléments d'information en plus

- Normalisation
  - La normalisation par couche est souvent utilisée. Elle aide l'apprentissage en standardisant la représentation de chaque mot (moyenne 0 et écart-type 1).
- Connexions résiduelles
  - $z_i = \text{Self-Attention}(X) + x_i$
  - Biaise vers des solutions simples, aide l'apprentissage
  - Utilisée pour le self-attention (figure à droite) & MLP



Qu’en est-il du décodage?



Qu’en est-il du décodage?

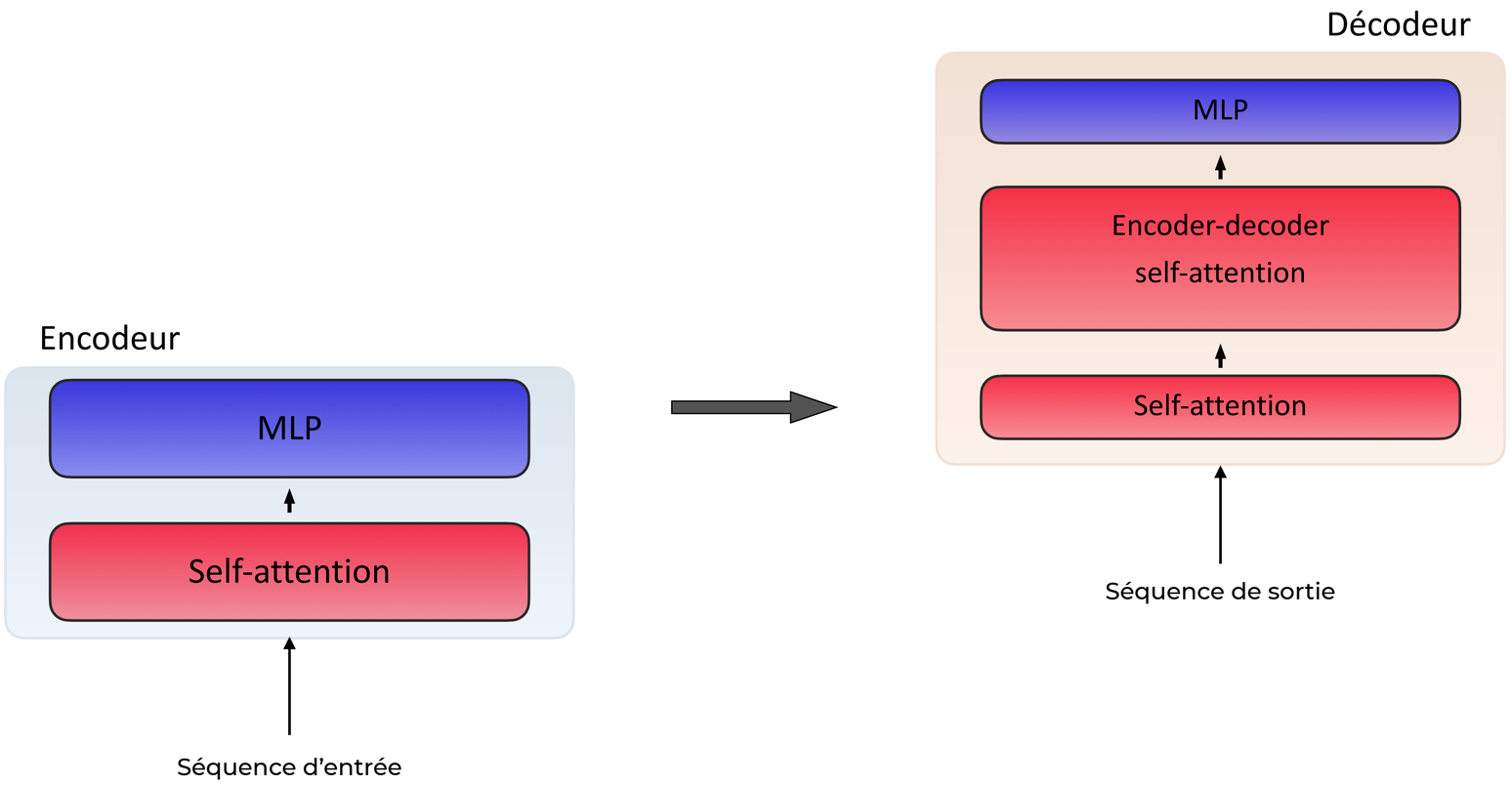
- Transformers transform word representations
  - Decoding requires using these words representations to obtain the following word
  - Add a “softmax”-layer at the end:

$$P(\mathbf{o_n} \mid \mathbf{o_{1:n-1}}) = \frac{\exp(\mathbf{g_w^\top x_{n-1}})}{\sum_{\mathbf{w}}^{\mathbf{W}} \exp(\mathbf{g_w^\top x_{n-1}})}$$

g_w: parameters
W: number of words in the vocabulary

  - You can only attend to previous words
    - Attention matrix is constrained to be (upper) triangular

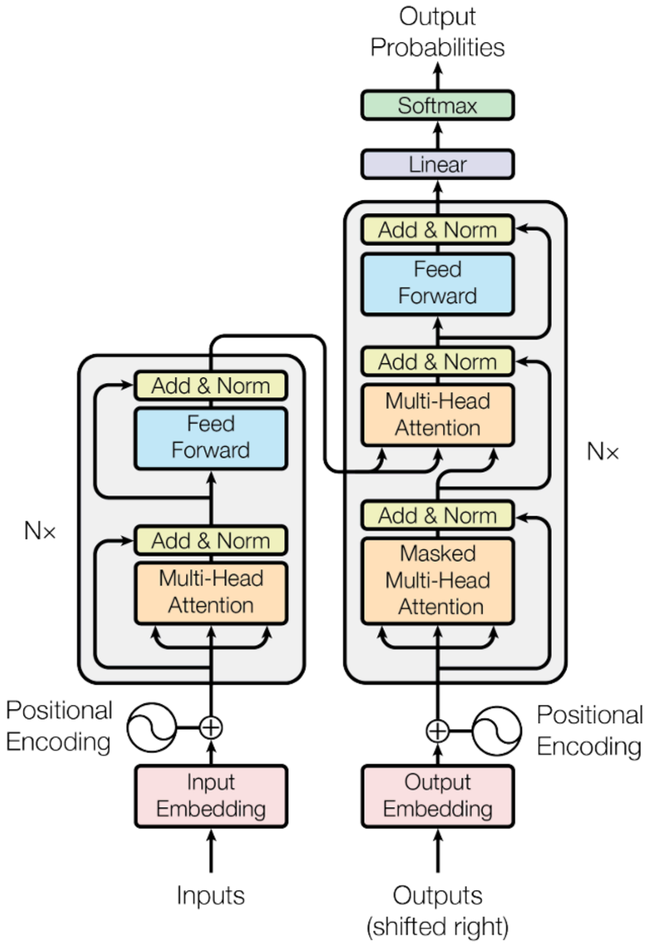
Transformeur Encodeur-Decodeur



On regroupe toutes ces notions

Attention Is All You Need, Vaswani et al.
https://arxiv.org/pdf/1706.03762

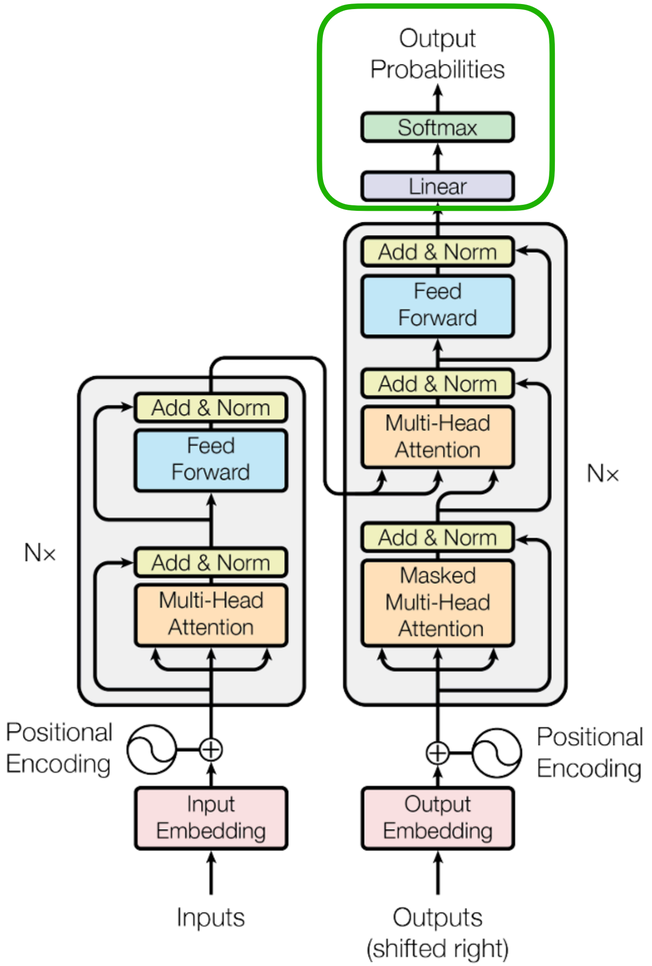
On regroupe toutes ces notions



- Self-Attention
- Positional embedding
- Encoder (N blocs)
- Decoder (N blocs)
- Softmax-layer
- Multiple transformer blocks (Nx)

Attention Is All You Need, Vaswani et al.
https://arxiv.org/pdf/1706.03762

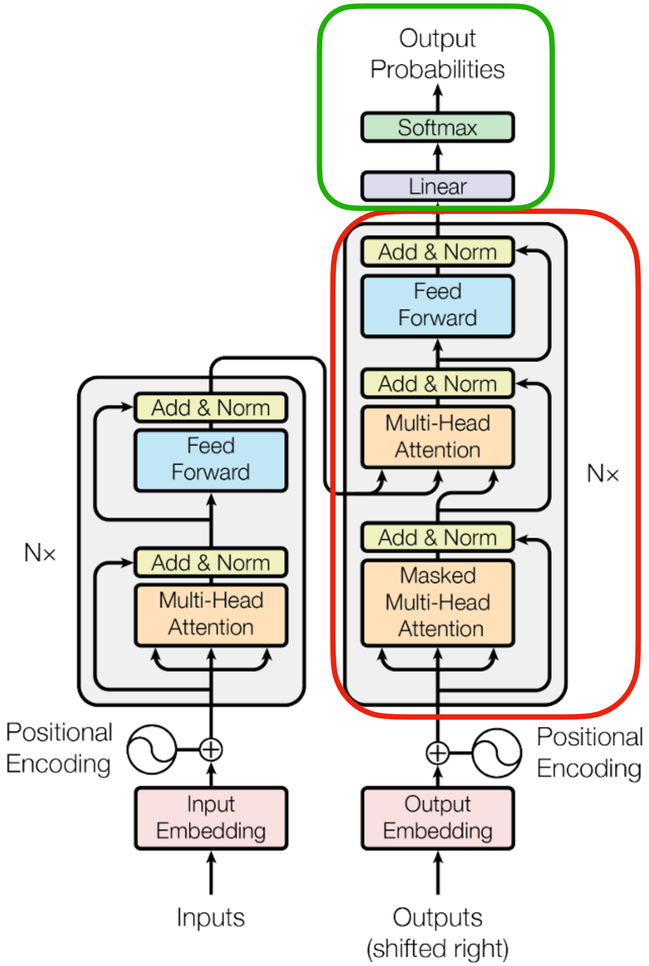
On regroupe toutes ces notions



- Self-Attention
- Positional embedding
- Encoder (N blocs)
- Decoder (N blocs)
- Softmax-layer
- Multiple transformer blocks (Nx)

Attention Is All You Need, Vaswani et al.
https://arxiv.org/pdf/1706.03762

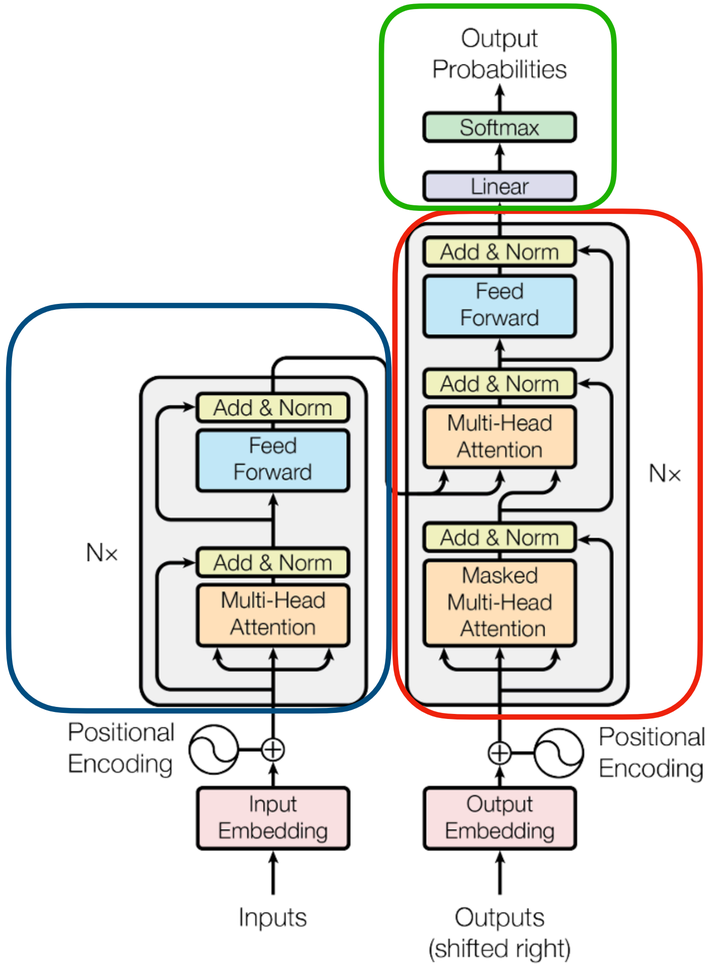
On regroupe toutes ces notions



- Self-Attention
- Positional embedding
- Encoder (N blocs)
- Decoder (N blocs)
- Softmax-layer
- Multiple transformer blocks (Nx)

Attention Is All You Need, Vaswani et al.
https://arxiv.org/pdf/1706.03762

On regroupe toutes ces notions



- Self-Attention
- Positional embedding
- Encoder (N blocs)
- Decoder (N blocs)
- Softmax-layer
- Multiple transformer blocks (Nx)

Attention Is All You Need, Vaswani et al.
https://arxiv.org/pdf/1706.03762

Objectif pour l’entraînement

- Souvent en deux (ou plus) étapes:
  - Première étape (non supervisée)
    - Prédiction du prochain mot dans une séquence
    - Prédiction d’un mot choisi aléatoirement dans la séquence (mot masqué)
    - D’autres tâches :
      - Prédiction de la prochaine phrase
  - Seconde étape (supervisée)
    - Préférences humaines ou une tâche supervisée

Il existe des « tas » de transformeurs

- Liste partielle: https://huggingface.co/docs/transformers/en/index
- Encodeur
  - BERT (RoBERTa), ALBERT
- Encoder-Decoder:
  - BART
- Décodeur:
  - Generative pre-trained transformer (GPT), BLOOM (for code), Llama
- *La majorité de ces transformeurs sont en fait des systèmes entraînés et pas seulement des architectures. Plusieurs de ces modèles ont maintenant des dizaines de milliards de paramètres (Llama-7B -> 7 milliards de paramètres)
- Aussi utilisés pour modéliser des images, de l’audio, multimodales, etc.

Sommaire

- Self-attention est l’ingrédient clé
  - Efficace et obtient de bons résultats empiriques
  - Les LLMs récent peuvent avec de très longues séquences d’entrée
  - Permets d’entraîner des transformeurs en utilisant beaucoup plus de données
- Les transformeurs sont maintenant « standard » et ils obtiennent souvent des performances supérieures aux RNNs/CNNs quand il existe suffisamment de données (biais inductif plus simple?)

Références

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