"Vous ne pouvez pas condenser la signification d'une phrase %&!$# dans un seul vecteur $&!#* !"
-- Ray Mooney, Association pour la linguistique informatique (ACL) 2014

Le Guide du voyageur galactique et la révolution de la fouille des textes

L'agenda d'aujourd'hui

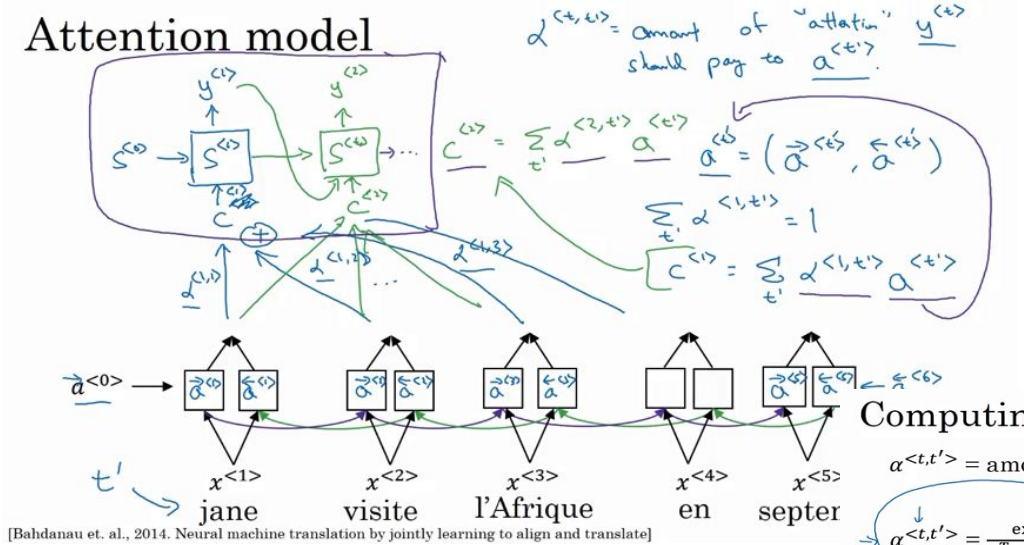
- Attention
  - Mise en œuvre de PyTorch
  - A-t-il un biais inductif ?
- Le modèle de transformer
  - L'architecture générale
  - L'attention à plusieurs têtes
  - Mise en œuvre de PyTorch
- L'avant-garde des RNN et des transformers
  - ELMo
  - BERT
  - GPT-2

l'attention

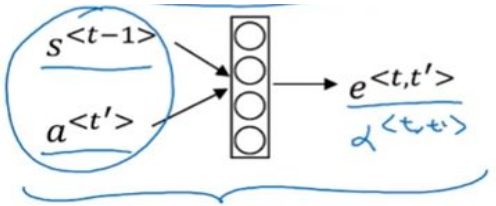
Liste de choses à régler dans le vieux TM

- La représentation des mots n'est pas sensible au contexte.
- Le traitement séquentiel des RNN est mauvais.
  - La langue n'est pas entièrement traitée de manière linéaire.
  - L’oublie des longues séquences (gradient disparaissant, même avec une porte oubliée).
  - La passe en avant n'est pas parallélisable.

Attention model



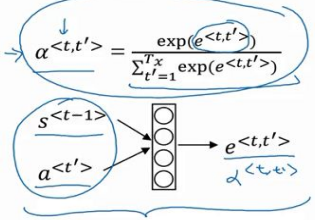
[Bahdanau et. al., 2014. Neural machine translation by jointly learning to align and translate]

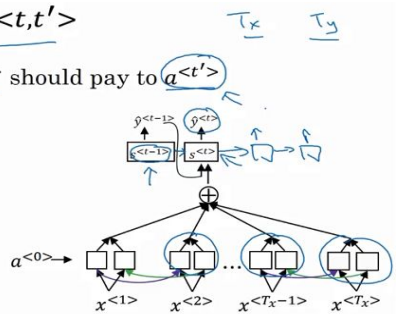


Computing attention $\alpha^{<t,t'>}$

$\alpha^{<t,t'>}$ = amount of attention $y^{<t>}$ should pay to $a^{<t'>}$

$$\alpha^{<t,t'>} = \frac{\exp(e^{<t,t'>})}{\sum_{t'=1}^{T_x} \exp(e^{<t,t'>})}$$





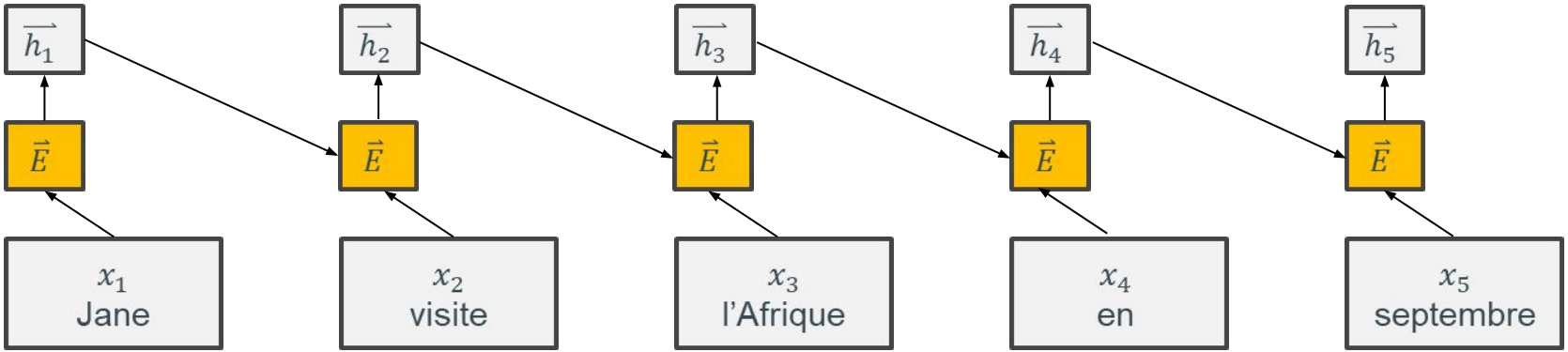
[Bahdanau et. al., 2014. Neural machine translation by jointly learning to align and translate]
[Xu et. al., 2015. Show, attend and tell: Neural image caption generation with visual attention]

Mettons l'accent sur un modèle de traduction seq2seq qui utilise un codeur biLSTM et un décodeur LSTM.

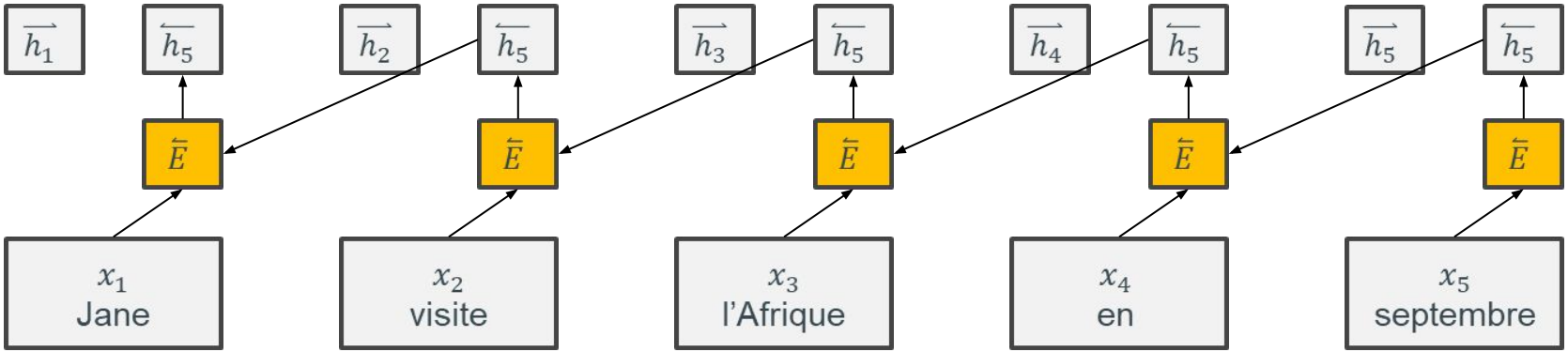
Traduisons du français à l'anglais.

x_1	x_2	x_3	x_4	x_5
Jane	visite	l'Afrique	en	septembre

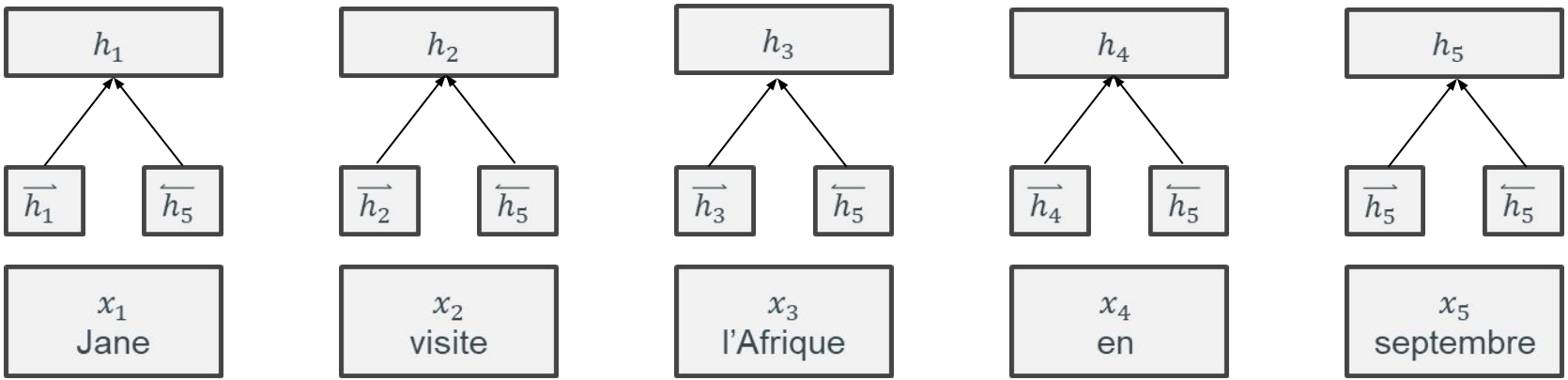
Tout d'abord, nous calculons les états cachés du biLSTM qui sont orientés vers l'avenir.



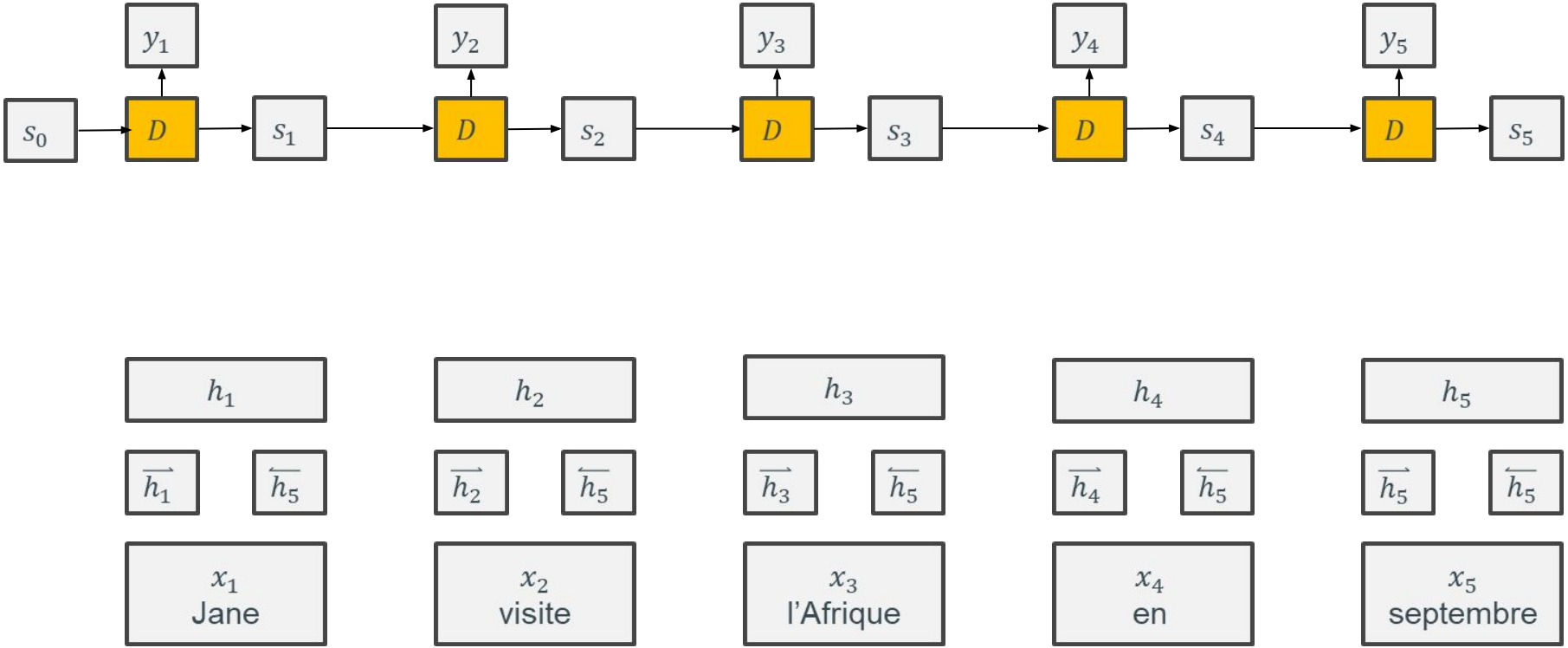
Ensuite, nous calculons les états cachés du biLSTM qui sont orientés vers l'arrière.



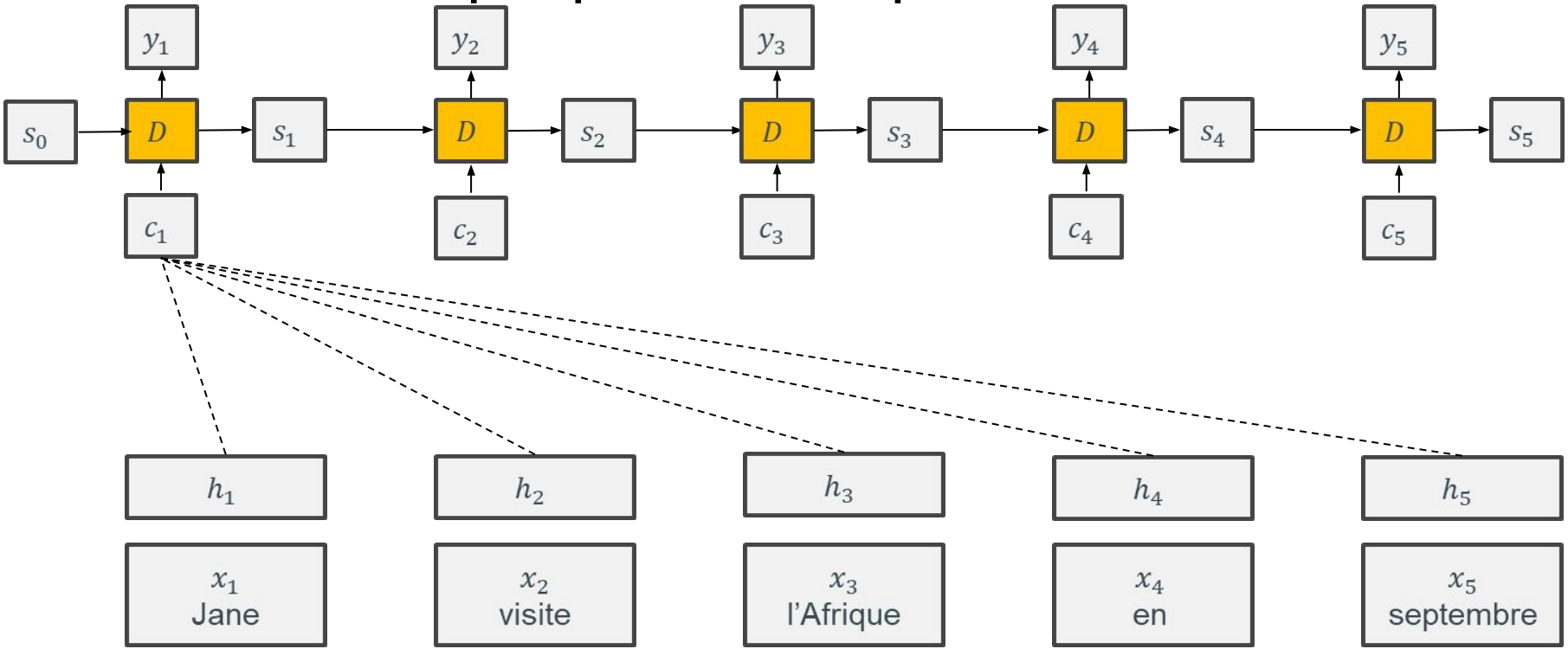
Nous concaténons ensuite nos états cachés.



Visualisez maintenant le décodeur.



Nous voulons qu'un vecteur de contexte soit introduit dans *D à* chaque pas de temps.



Comment coder les informations pertinentes provenant de *n* états cachés sur un vecteur de contexte de longueur fixe ?

Comment coder les informations pertinentes provenant de *n* états cachés sur un vecteur de contexte de longueur fixe ?

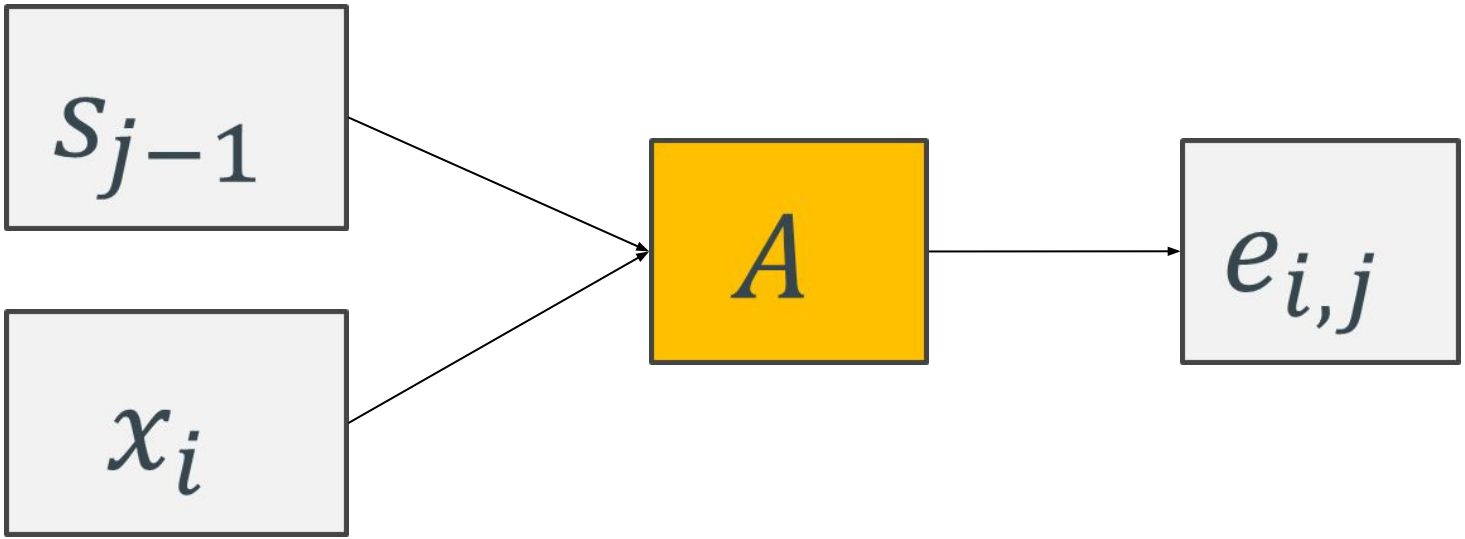
Pour la variable *n*.

Comment coder les informations pertinentes provenant de *n* états cachés sur un vecteur de contexte de longueur fixe ?

Pour la variable *n*.

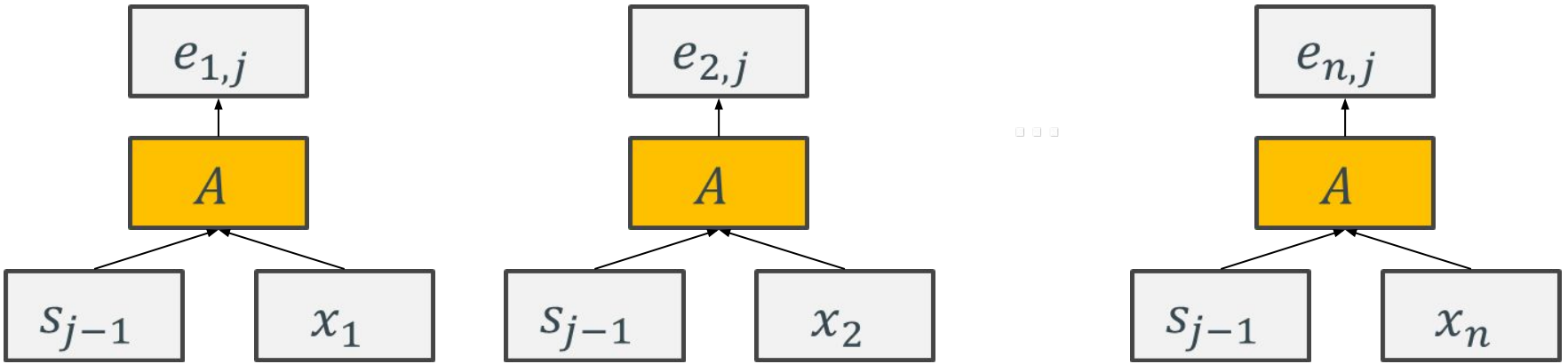
Pour une étape spécifique *i*.

Attention's approach: handle one input timestep (i) and one

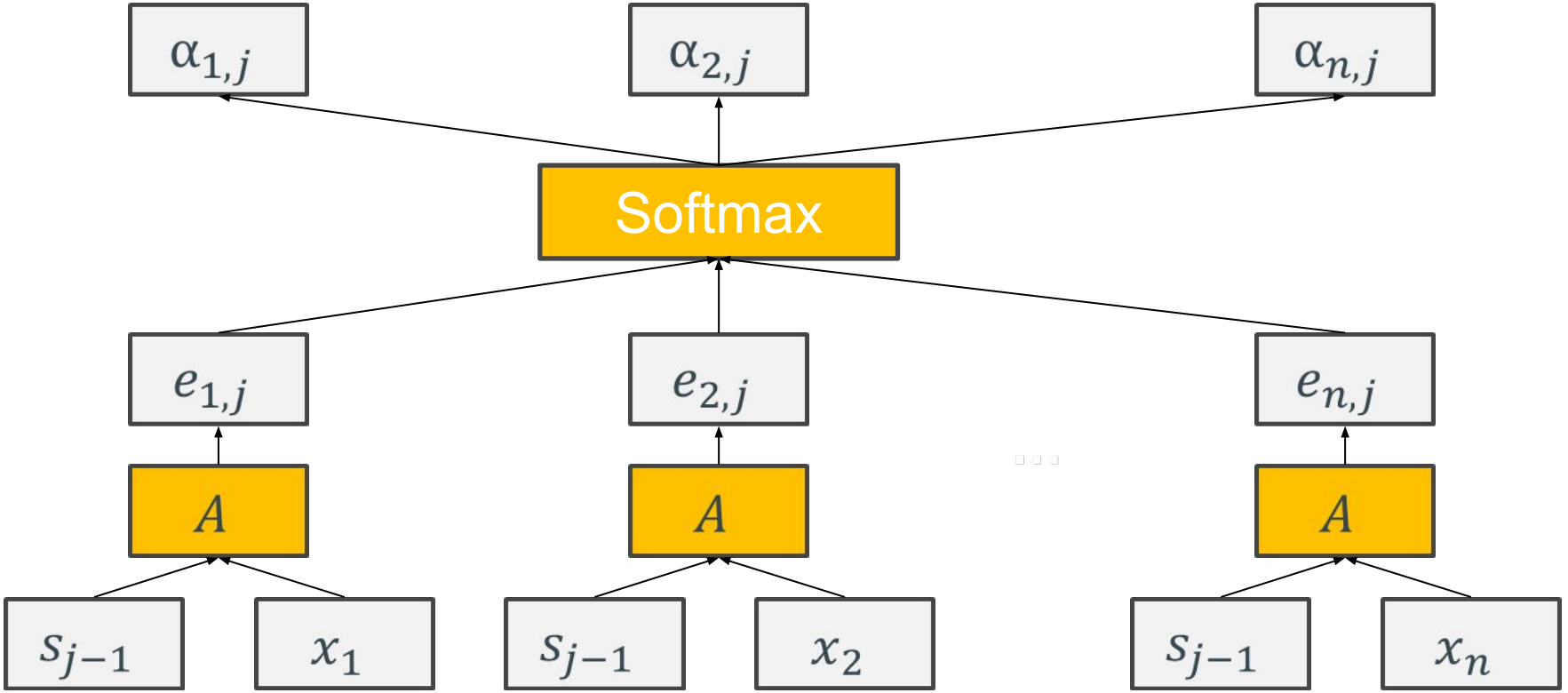


$e_{i,j}$ estimates how relevant x_i is to D during timestep j.

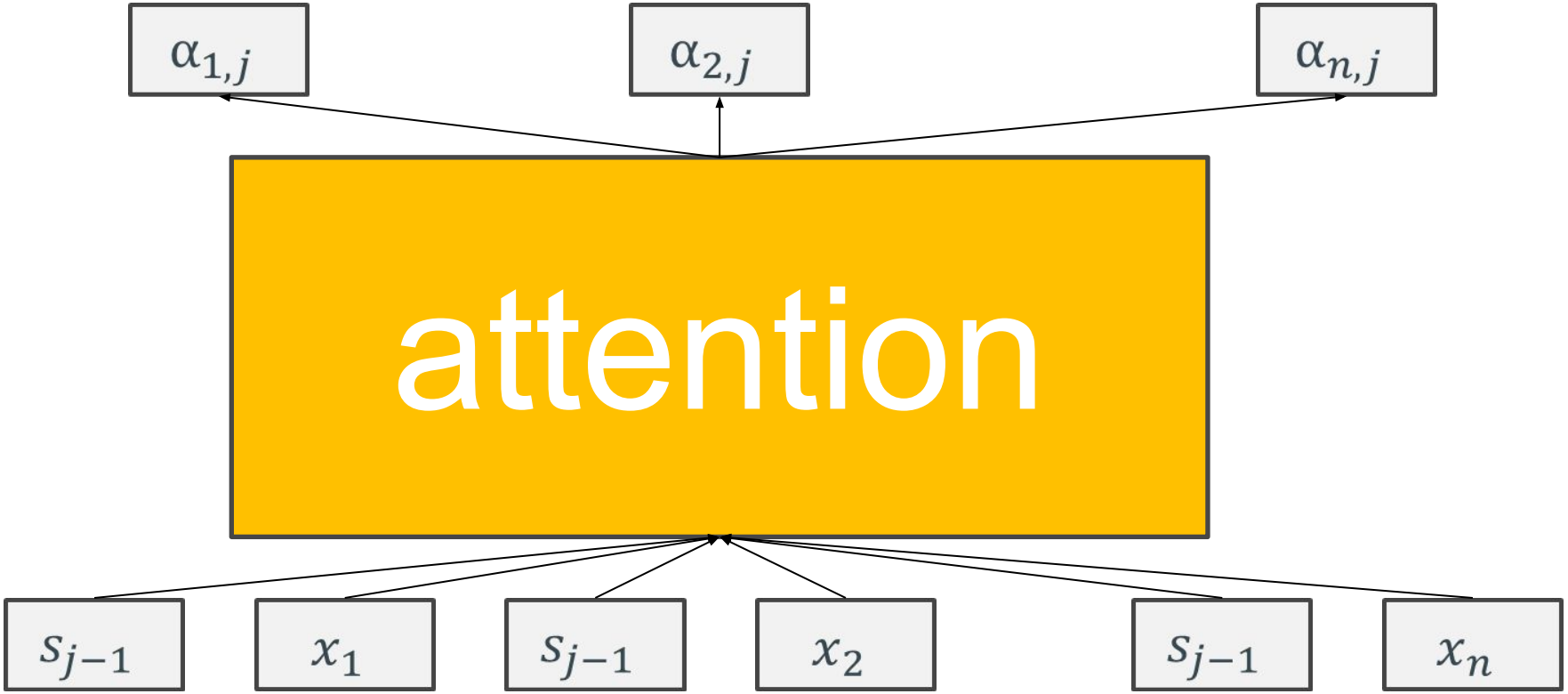
For timestep j, we compute all of the attention weights...



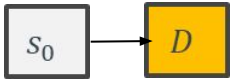
...et softmax.

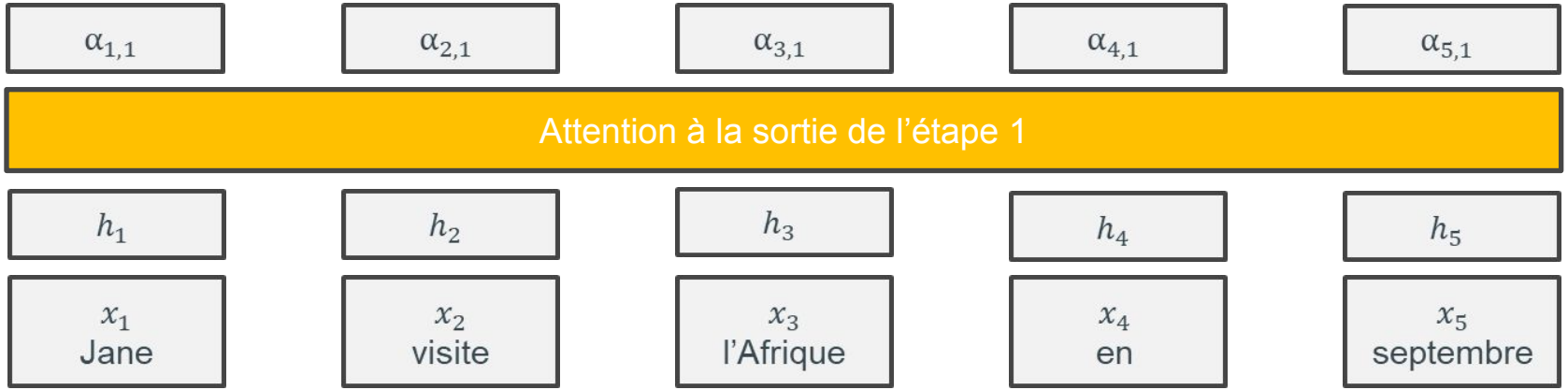

$\alpha_{1,j}$
$\alpha_{2,j}$
$\alpha_{n,j}$
Softmax
$e_{1,j}$
$e_{2,j}$
$e_{n,j}$
A
A
A
s_{j-1}
x_1
s_{j-1}
x_2
s_{j-1}
x_n

Nous l'appellerons le module d'attention.

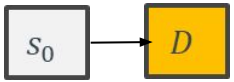


Le module d'attention nous donne un poids pour chaque entrée.

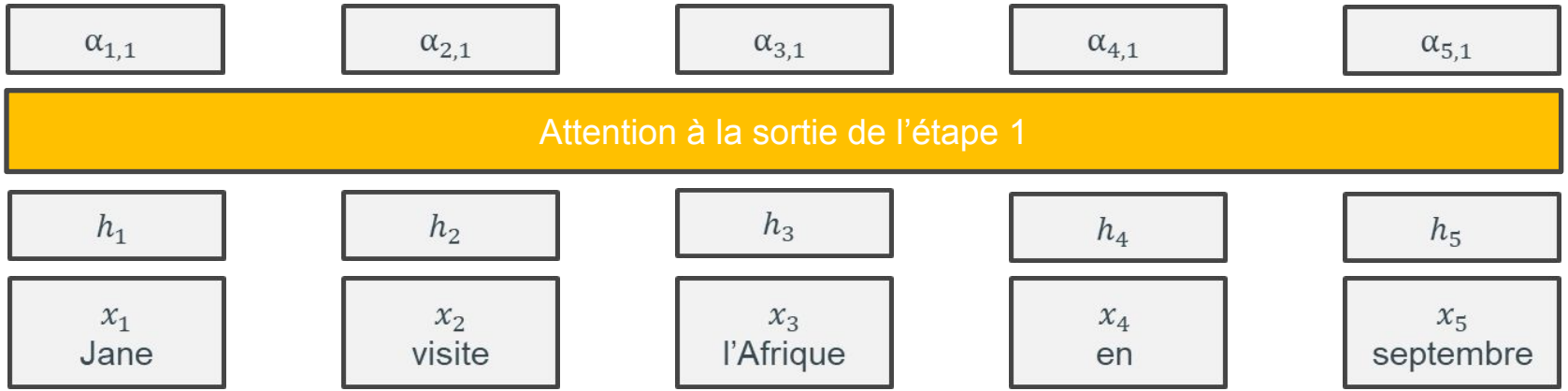




Le vecteur de contexte est une somme pondérée des encodages cachés.

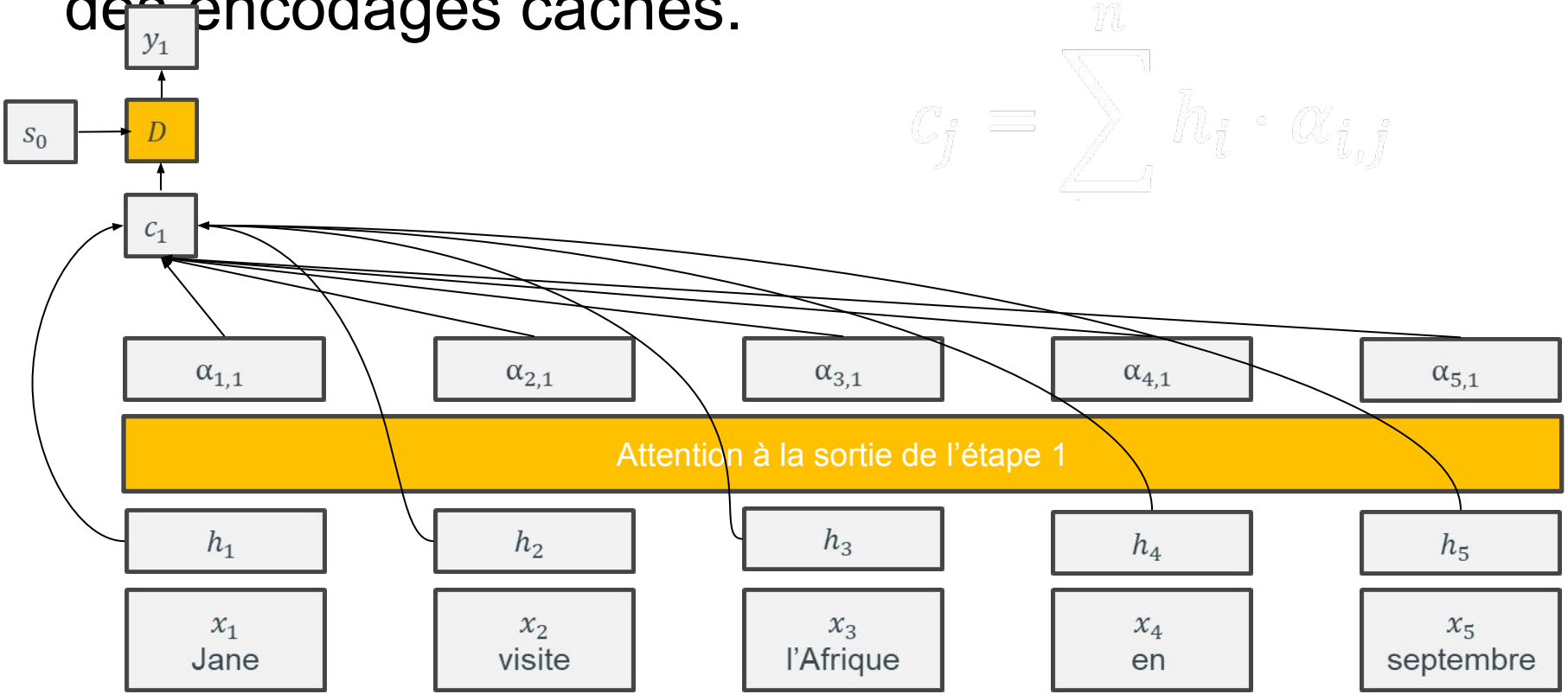


$$c_j = \sum_{i}^{n} h_i \cdot \alpha_{i,j}$$

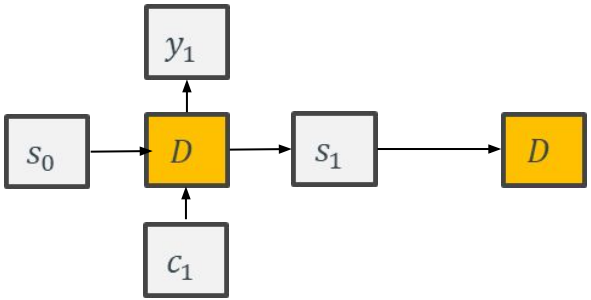


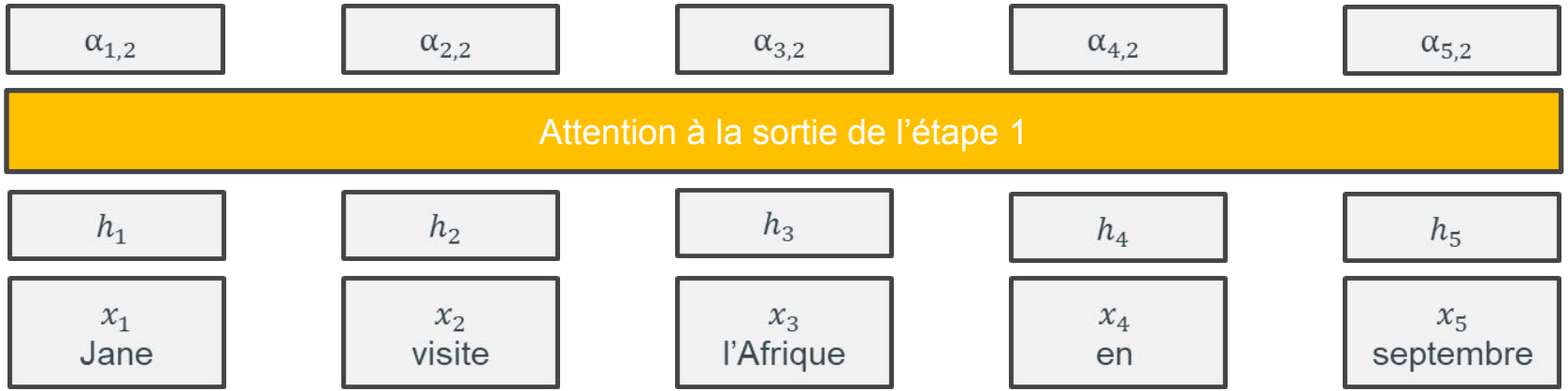
Le vecteur de contexte est une somme pondérée des encodages cachés.

$$c_j = \sum_{i}^{n} h_i \cdot \alpha_{i,j}$$

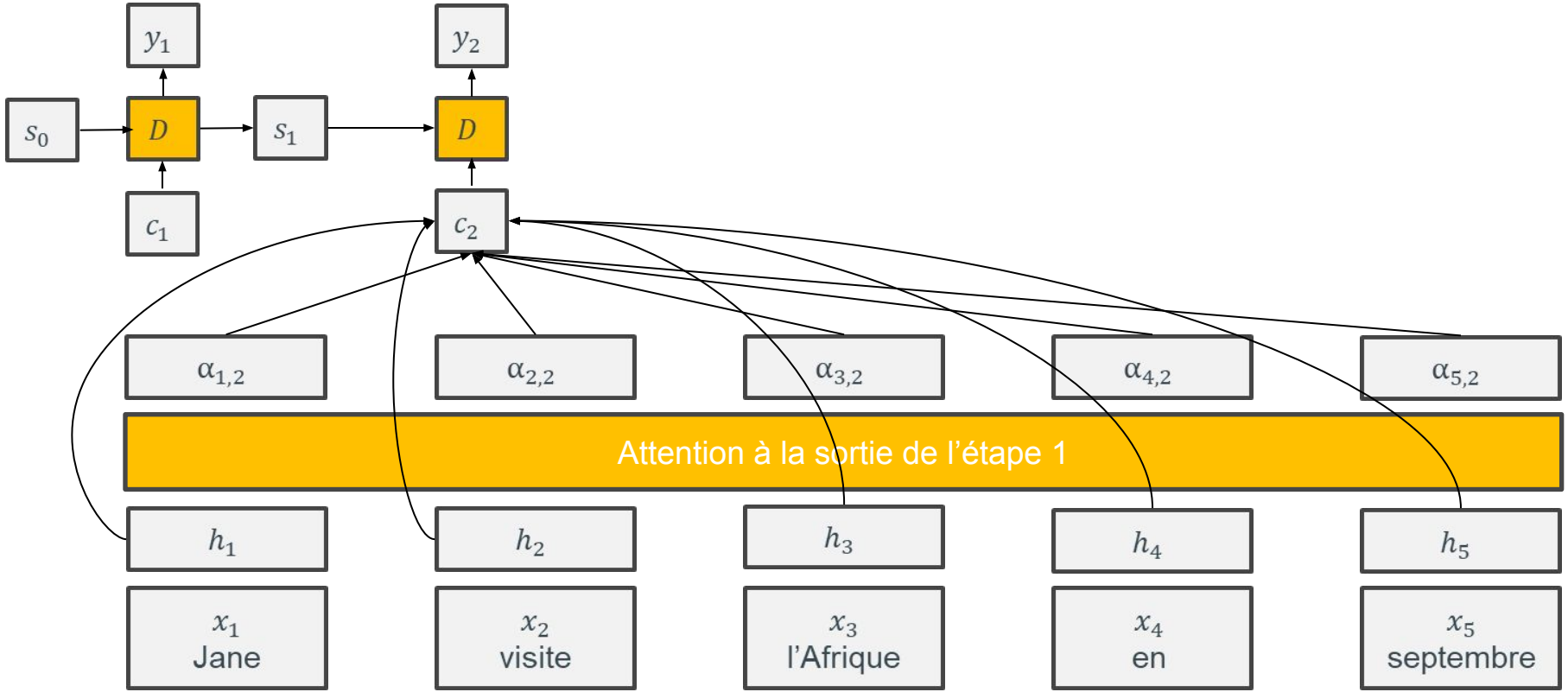


Nous répétons ensuite pour les étapes suivantes.

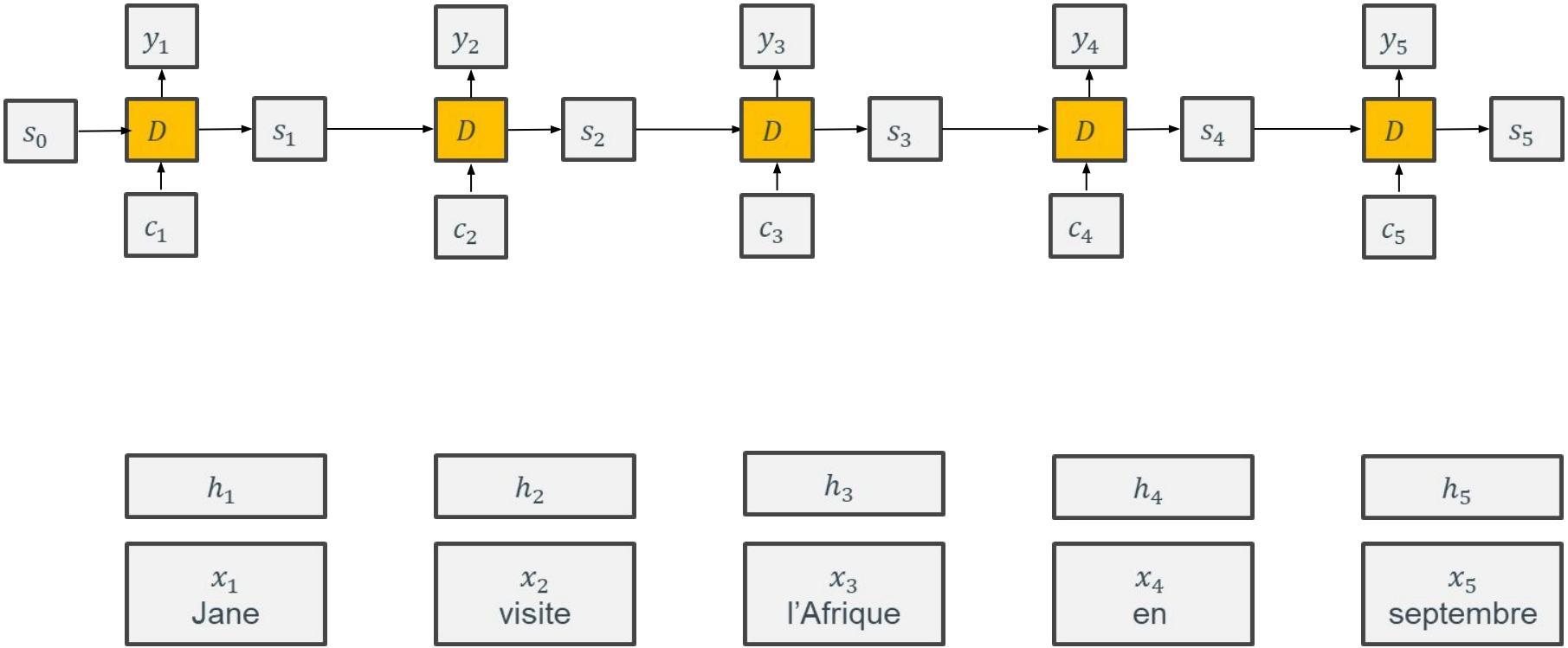




Nous répétons ensuite pour les étapes suivantes.



C'est fait !


y_1
y_2
y_3
y_4
y_5
s_0
D
s_1
D
s_2
D
s_3
D
s_4
D
s_5
c_1
c_2
c_3
c_4
c_5
h_1
h_2
h_3
h_4
h_5
x_1 Jane
x_2 visite
x_3 l'Afrique
x_4 en
x_5 septembre

Ce type d'architecture est très performant (SOTA vers 2016).

German-English translations	
src	In einem Interview sagte Bloom jedoch , dass er und Kerr sich noch immer lieben .
ref	However , in an interview , Bloom has said that he and *Kerr* still love each other .
best	In an interview , however , Bloom said that he and *Kerr* still love .
base	However , in an interview , Bloom said that he and **Tina** were still <unk> .
src	Wegen der von Berlin und der Europäischen Zentralbank verhängten strengen Sparpolitik in Verbindung mit der Zwangsjacke , in die die jeweilige nationale Wirtschaft durch das Festhalten an der gemeinsamen Währung genötigt wird , sind viele Menschen der Ansicht , das Projekt Europa sei zu weit gegangen
ref	The *austerity imposed by Berlin and the European Central Bank , coupled with the straitjacket* imposed on national economies through adherence to the common currency , has led many people to think Project Europe has gone too far .
best	Because of the strict *austerity measures imposed by Berlin and the European Central Bank in connection with the straitjacket* in which the respective national economy is forced to adhere to the common currency , many people believe that the European project has gone too far .
base	Because of the pressure **imposed by the European Central Bank and the Federal Central Bank with the strict austerity** imposed on the national economy in the face of the single currency , many people believe that the European project has gone too far .

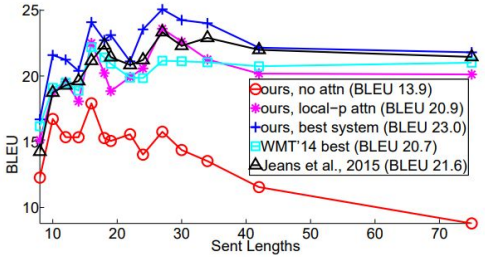


Figure 6: **Length Analysis** – translation qualities of different systems as sentences become longer.

System	Ppl	BLEU	
		Before	After unk
global (location)	6.4	18.1	19.3 (+1.2)
global (dot)	6.1	18.6	20.5 (+1.9)
global (general)	6.1	17.3	19.1 (+1.8)
local-m (dot)	>7.0	x	x
local-m (general)	6.2	18.6	20.4 (+1.8)
local-p (dot)	6.6	18.0	19.6 (+1.9)
local-p (general)	**5.9**	**19**	**20.9 (+1.9)**

Table 4: **Attentional Architectures** – performances of different attentional models. We trained two local-m (dot) models; both have ppl > 7.0.

L'attention a-t-elle un biais inductif ?

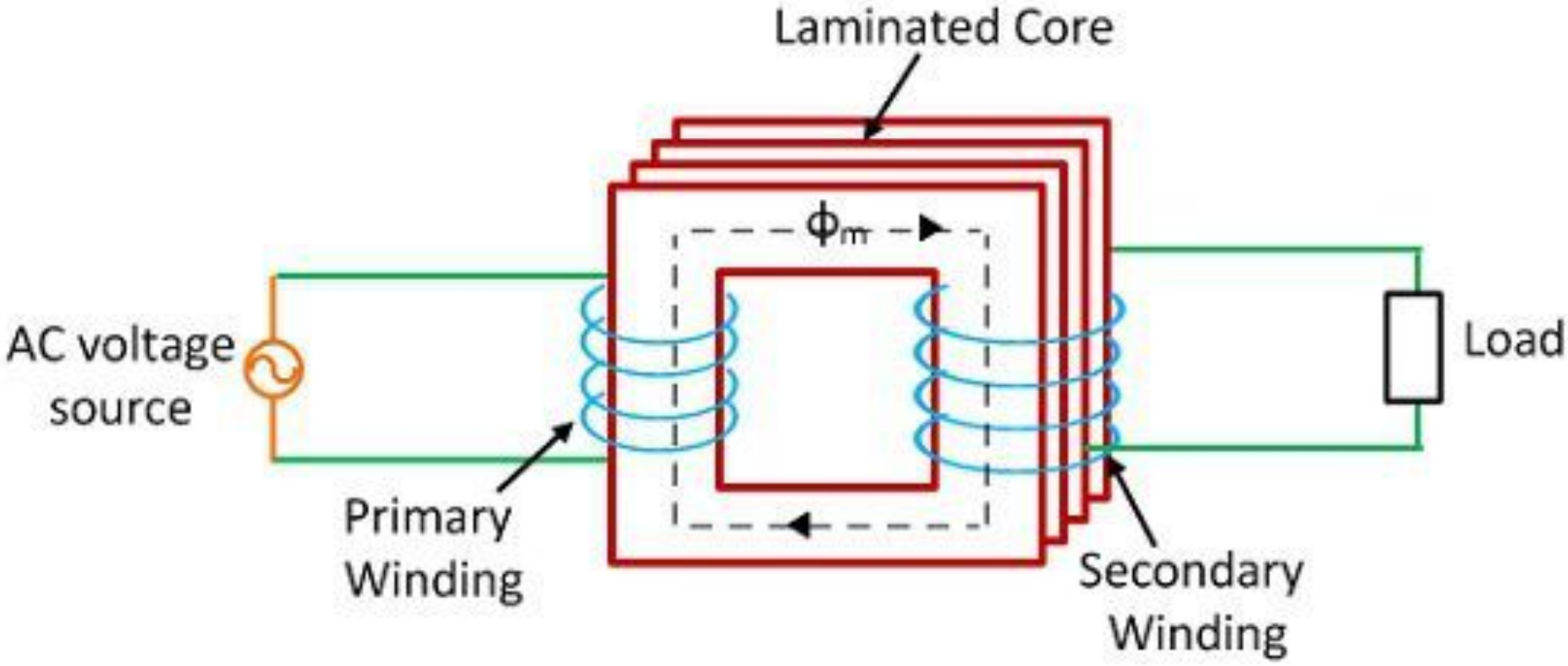
Interpréter l'attention comme un biais inductif

- La régularisation de la L1 confère une rareté préalable au niveau des params.
- L'attention (dure) confère une rareté préalable au niveau du calcul/flux d'informations.
- Attention douce : attention dure : : Régularisation L2 : Régularisation de la L1.

Qu'est-ce qui ne va pas avec les modèles de seq2seq attentionnés ?

- Le codeur (facultatif) est toujours un RNN.
  - Et vous n'irez pas très loin sans.
- Le décodeur (non négociable) est toujours un RNN.

Transformateurs



Electrical Transformer

Qu'est-ce qu'un transformer ?

- Un modèle de Google Brain.
  - Entrée de longueur variable
  - Sortie de longueur fixe (mais généralement étendue à une sortie de longueur variable)
  - **Pas de récurrence**
  - Étonnamment, il n'est pas breveté.
- Utilise 3 types d'attention
  - L'auto-attention du codeur.
  - L'auto-attention du décodeur.
  - Encodeur-décodeur attention multi-tête.

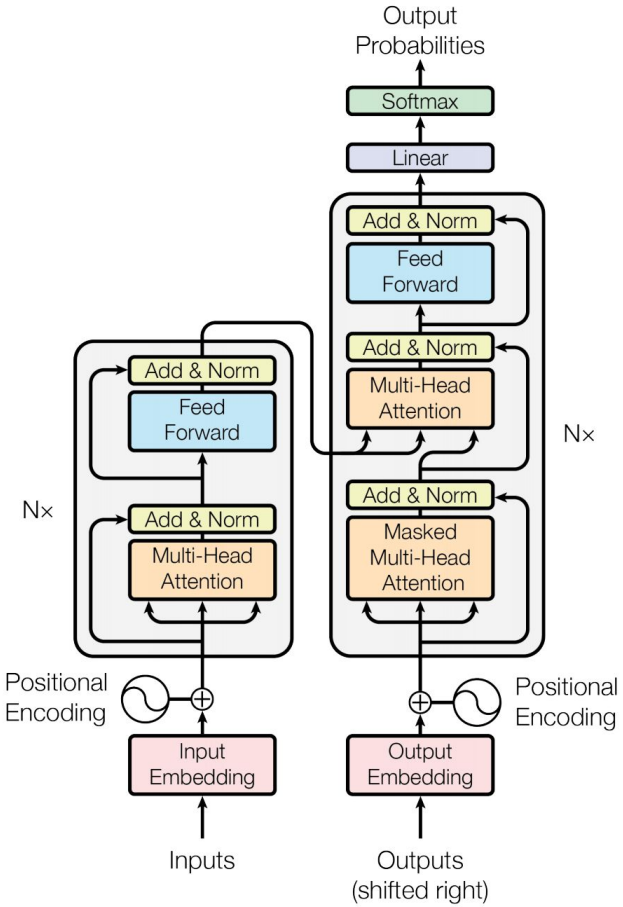
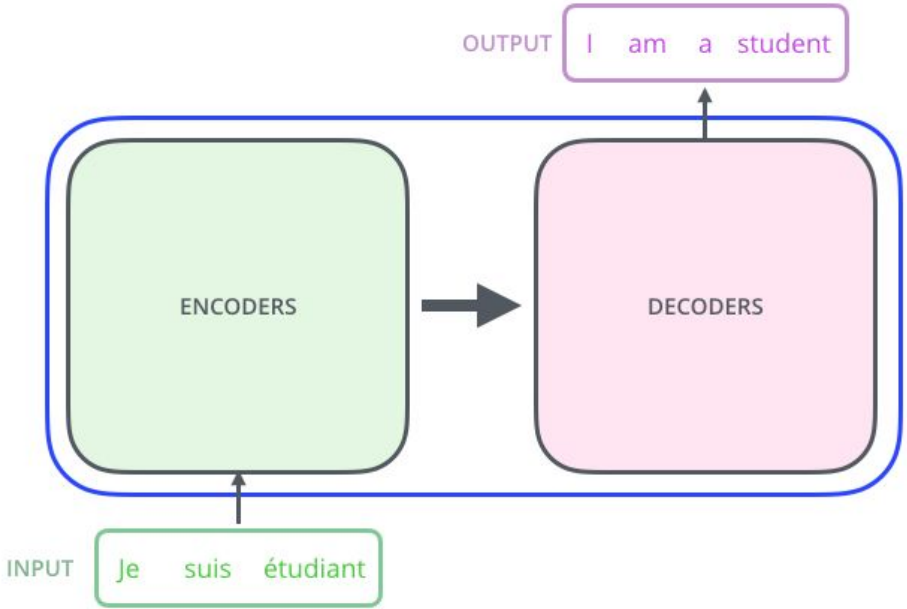


Figure 1: The Transformer - model architecture.

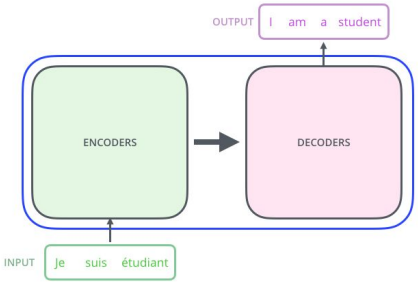
L'architecture générale

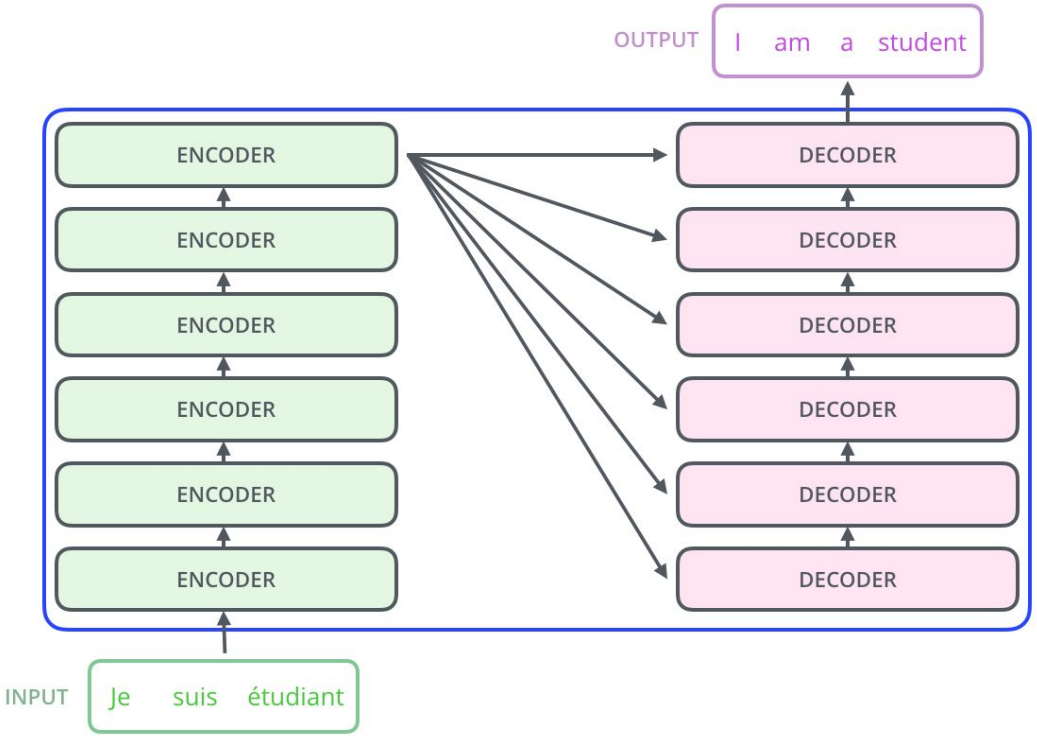
Construction d’un transformer étape par étape

Il y a un codeur et un décodeur.

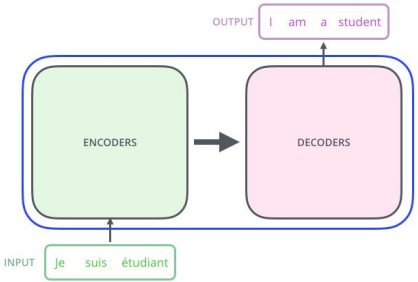


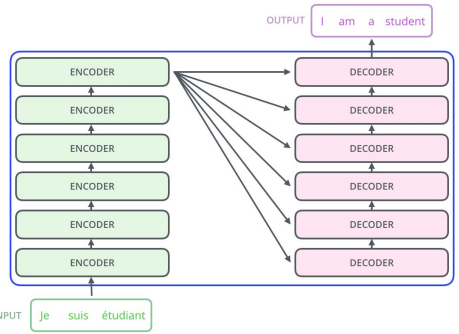
Il existe plusieurs couches de codage et de décodage.

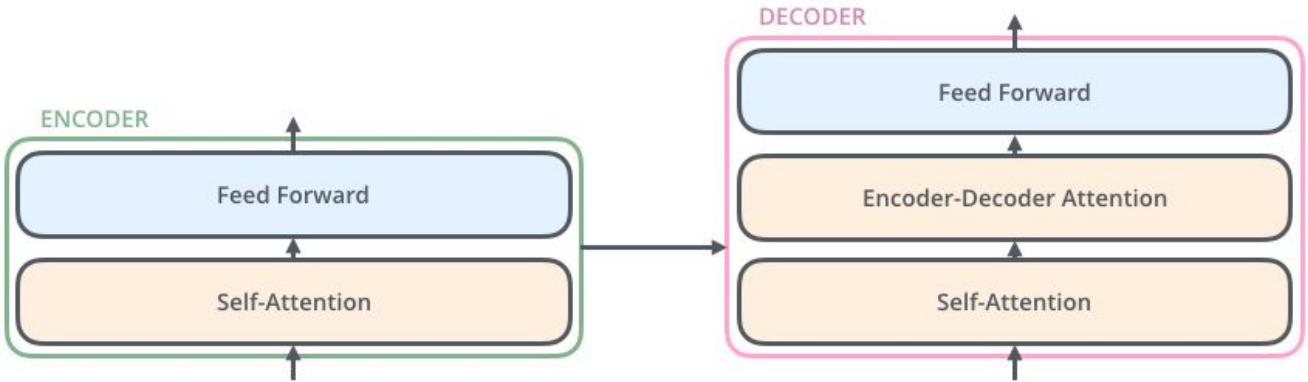




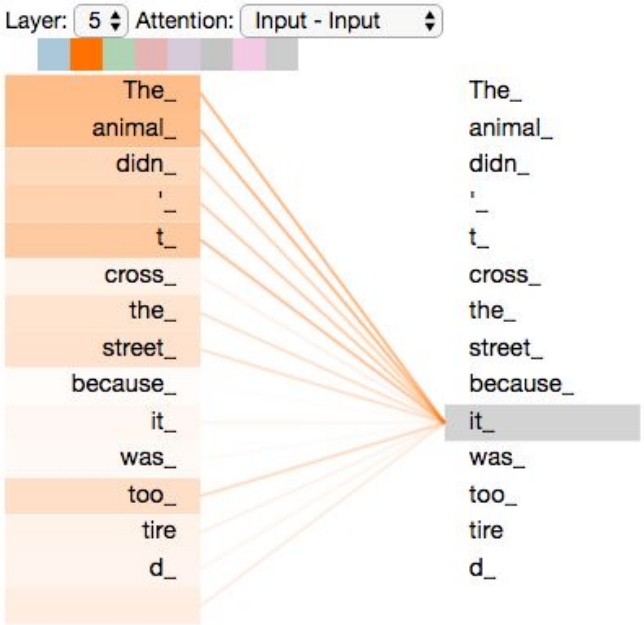
Chaque couche de codage / décodage possède une couche d'auto-attention et une ff.



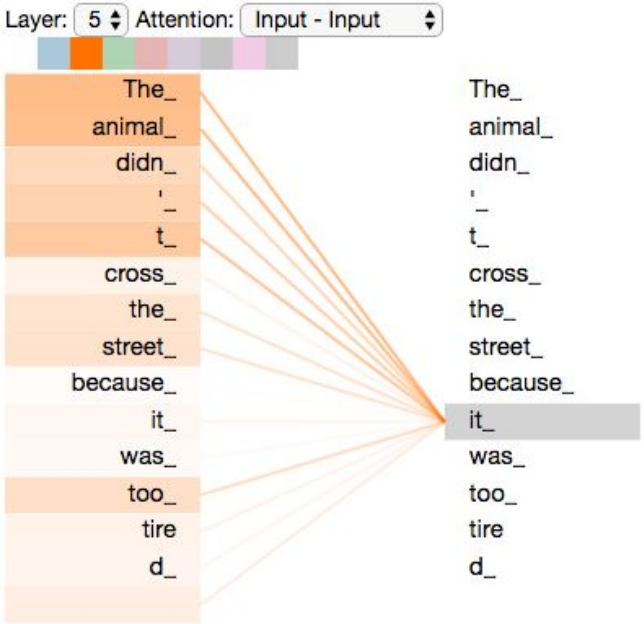




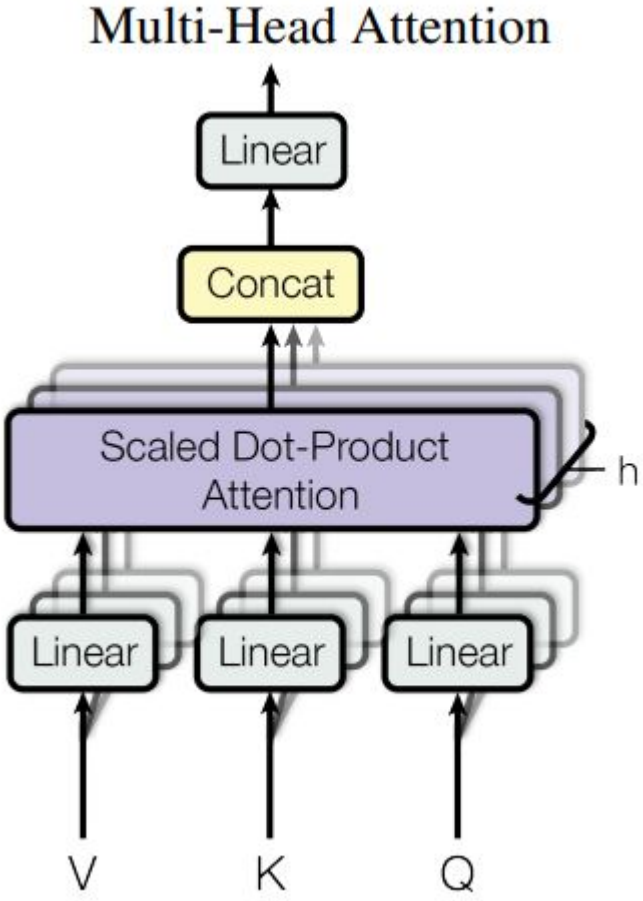
Les transformers utilisent l'auto-attention du codeur et du décodeur pour apprendre à mieux représenter les entrées en fonction du contexte.



Les transformers utilisent l'auto-attention du codeur et du décodeur pour apprendre à mieux représenter les entrées en fonction du contexte.



Comprendre l'attention multi-tête


Multi-Head Attention
Linear
Concat
Scaled Dot-Product Attention
h
Linear
Linear
Linear
V
K
Q

Les entrées : *K, V,* Q.

- Dans seq2seq attention, nous avons utilisé les *n* états cachés de l'encodeur pour apprendre les poids d'attention sur eux-mêmes (les états cachés de l'encodeur) afin de mettre à jour l'état caché du décodeur.
- Renommez-les comme clé, valeur, requête.
- Les vecteurs clés déterminent les poids d'attention.
- Les vecteurs de valeur sont multipliés par les poids d'attention.
- Les vecteurs de requête sont ce pour quoi nous essayons d'apprendre le contexte.

Attention aux produits à l'échelle du point

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

- Les vecteurs clés déterminent les poids d'attention.
- Les vecteurs de valeur sont multipliés par les poids d'attention.
- Les vecteurs de requête sont ce pour quoi nous essayons d'apprendre le contexte.

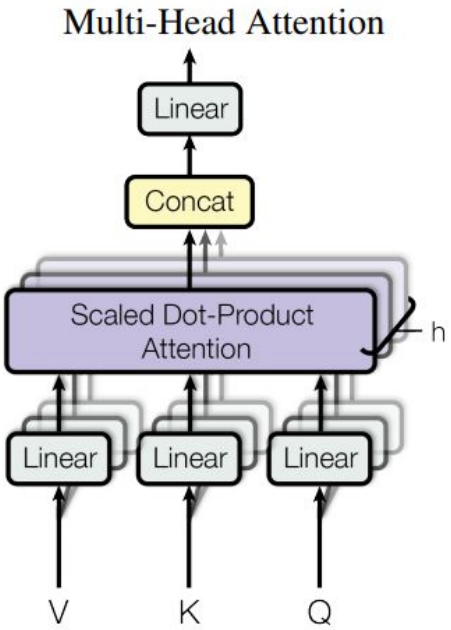
Attention aux produits à l'échelle du point

Nous pondérons chaque vecteur de valeur *v en fonction de* la similarité de sa clé correspondante avec la requête.

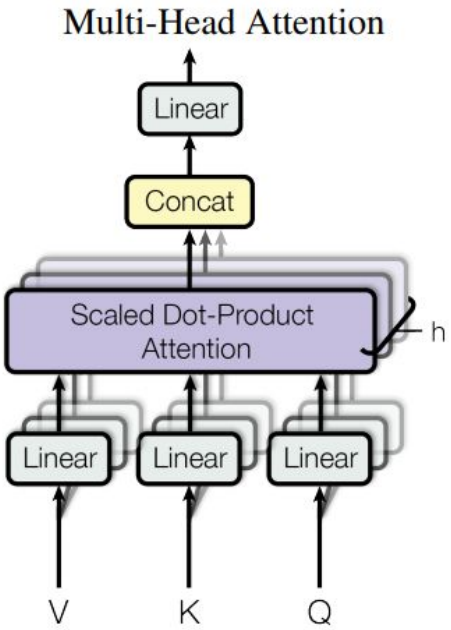
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

- Les vecteurs clés déterminent les poids d'attention.
- Les vecteurs de valeur sont multipliés par les poids d'attention.
- Les vecteurs de requête sont ce pour quoi nous essayons d'apprendre le contexte.

L'ensemble du module d'attention multi-tête

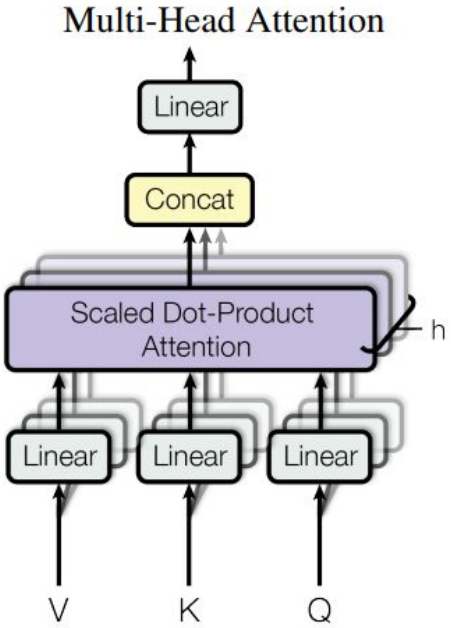


L'ensemble du module d'attention multi-tête



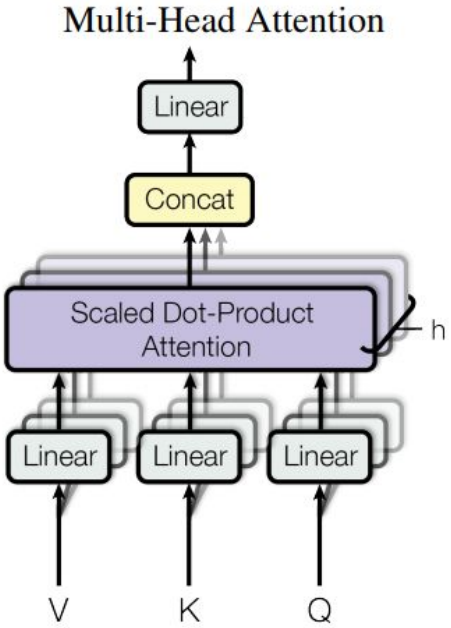
Prenez *h* transformations linéaires différentes de chaque entrée (*h* = 8 dans le document original).

L'ensemble du module d'attention multi-tête



Passez chacun des *h* entrées transformées triples ordonnées dans le module de produit à l'échelle du point.
Prenez *h* transformations linéaires différentes de chaque entrée (*h* = 8 dans le document original).

L'ensemble du module d'attention multi-tête

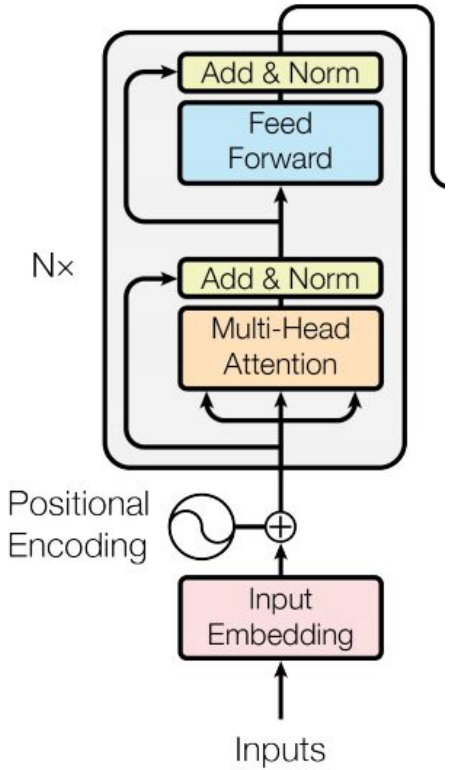


Une transformation linéaire de plus pour faire bonne métrique.

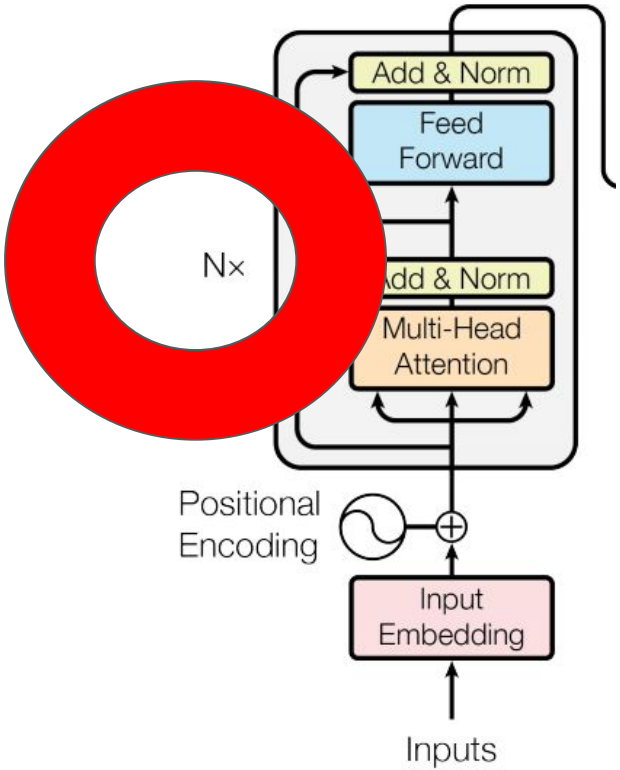
Passez chacun des *h* entrées transformées triples ordonnées dans le module de produit à l'échelle du point.

Prenez *h* transformations linéaires différentes de chaque entrée (*h* = 8 dans le document original).

Que sont les *K, V, Q* dans le codeur ?



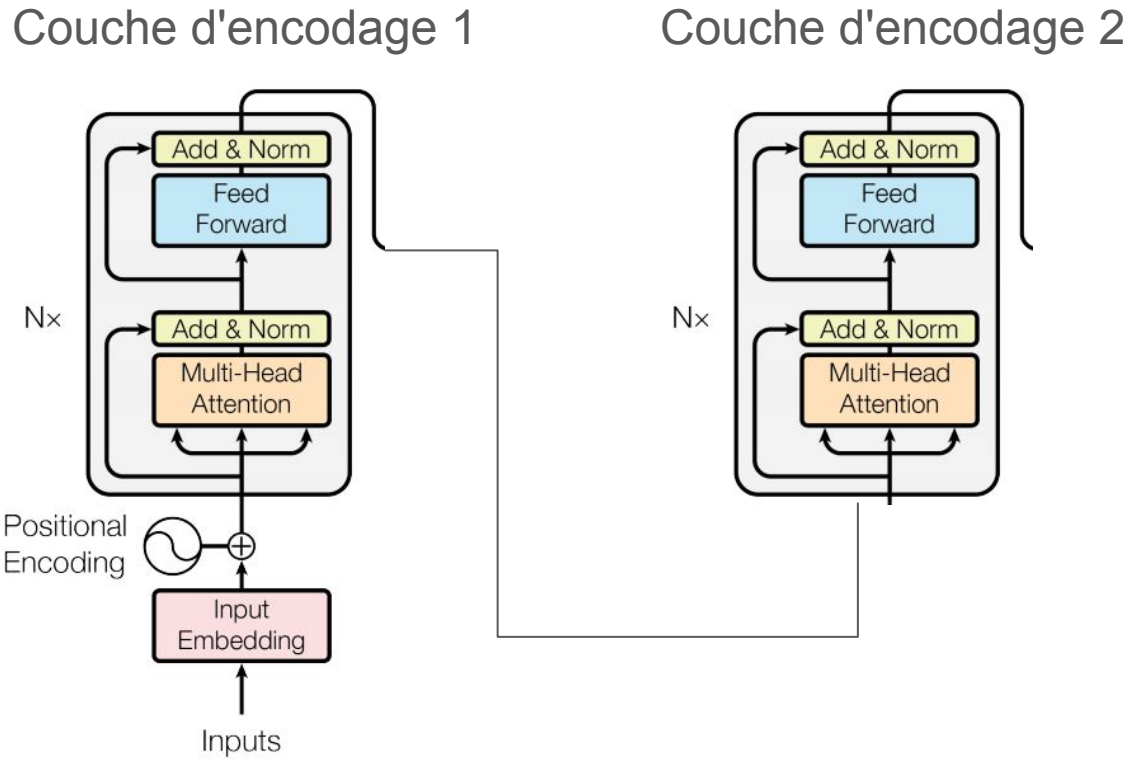
Que sont les *K, V, Q* dans le codeur ?



Que sont les *K, V, Q* dans le codeur ?

La requête est également l'entrée codée en fonction de la position.

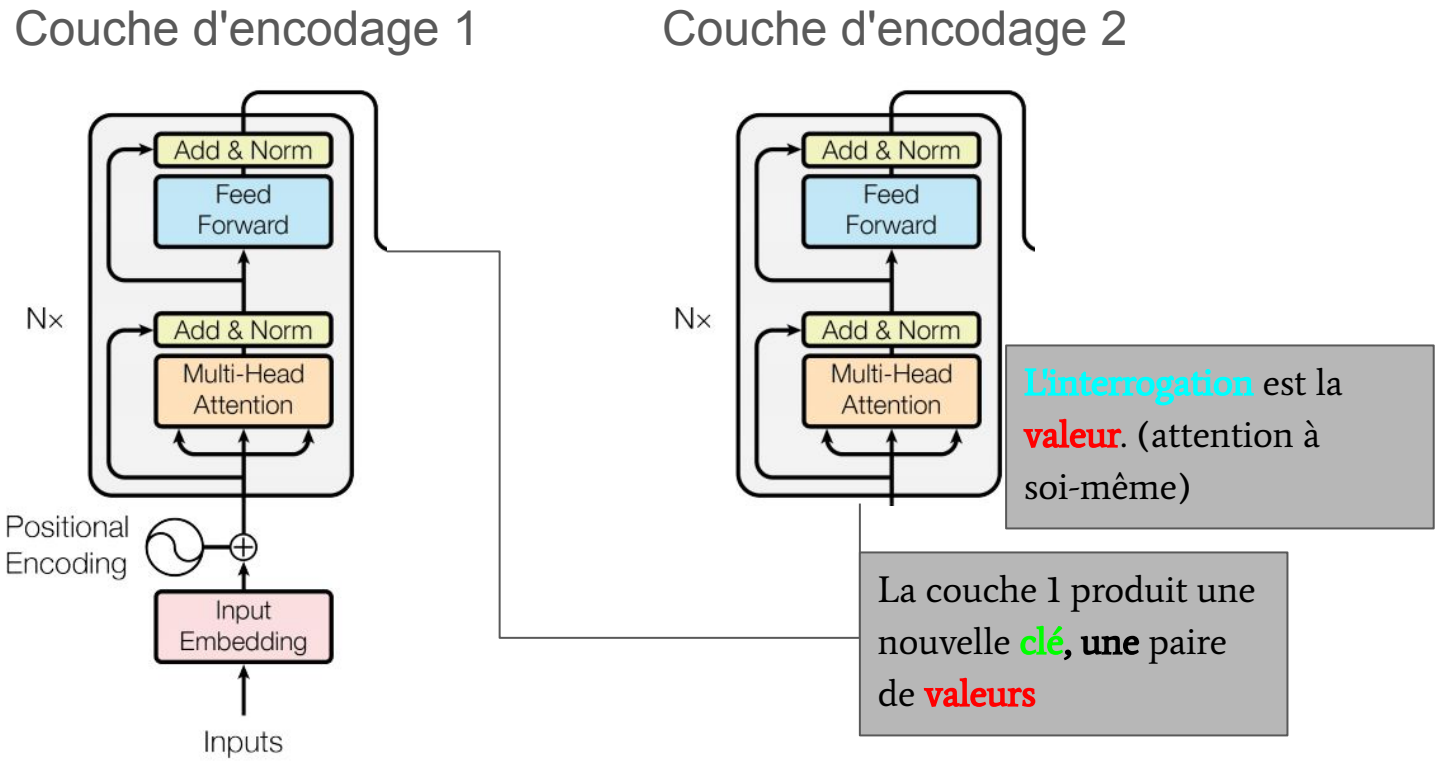
La clé et la **valeur** sont toutes deux des données codées en fonction de la position.



Que sont les *K*, *V*, *Q* dans le codeur ?

La requête est également l'entrée codée en fonction de la position.

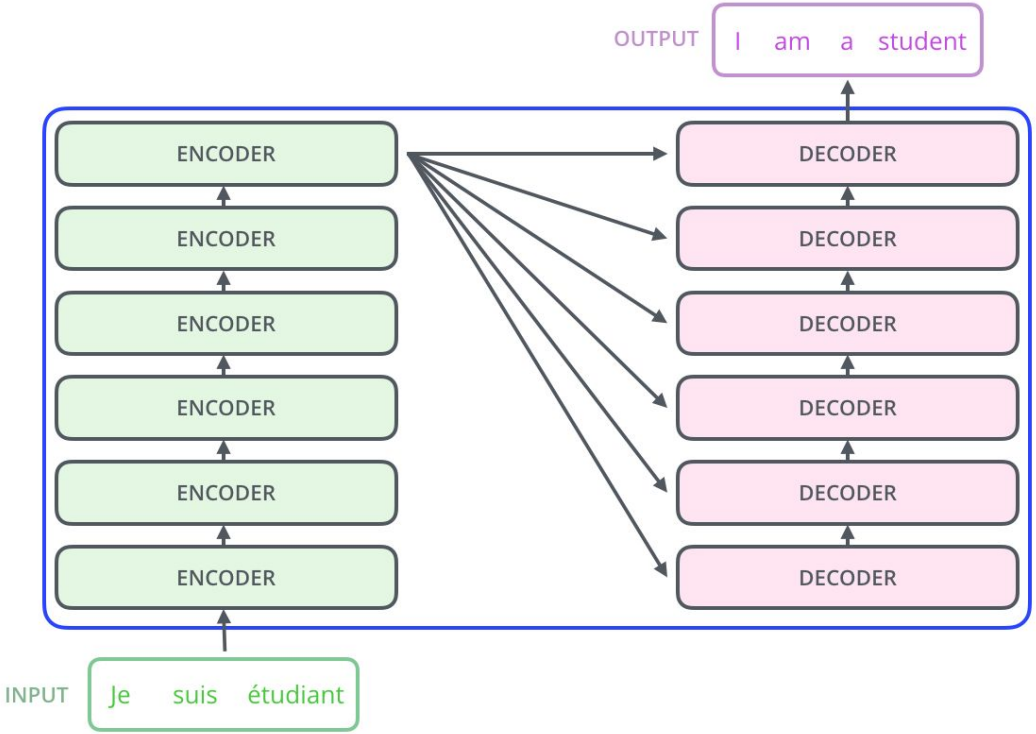
La clé et la valeur sont toutes deux des entrées codées en fonction de la position.



L'interrogation est la **valeur**. (attention à soi-même)

La couche 1 produit une nouvelle **clé**, **une** paire de **valeurs**

Le codeur envoie sa sortie finale (K, V) à chacune des couches du décodeur.



Que sont les *K*, *V*, *Q* dans le décodeur ?

Encodeur-décodeur attention multi-tête

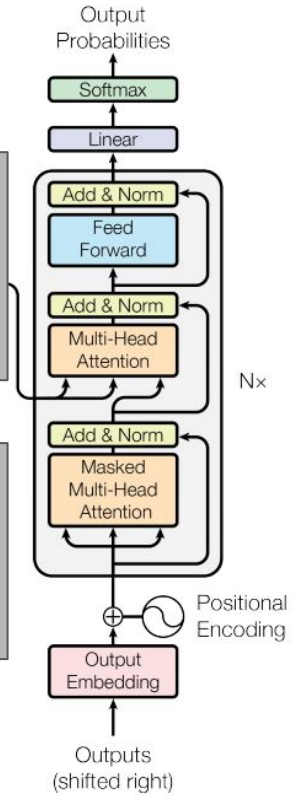
La clé, **la valeur** sont les sorties de la couche d'encodage finale.

La requête est la sortie de la couche décodeur précédente.

L'auto-attention du décodeur

La clé, **la valeur** sont les résultats de la couche précédente.

L'interrogation est la **valeur** (attention à soi-même).



Un dernier regard sur l'ensemble de l'architecture.

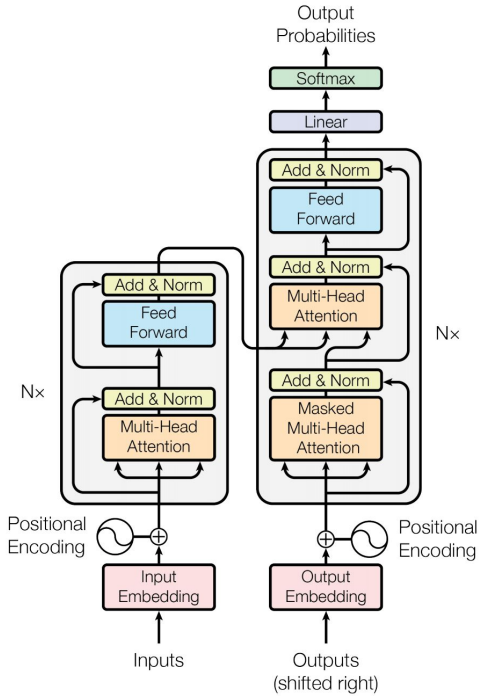


Figure 1: The Transformer - model architecture.

Un dernier regard sur l'ensemble de l'architecture.

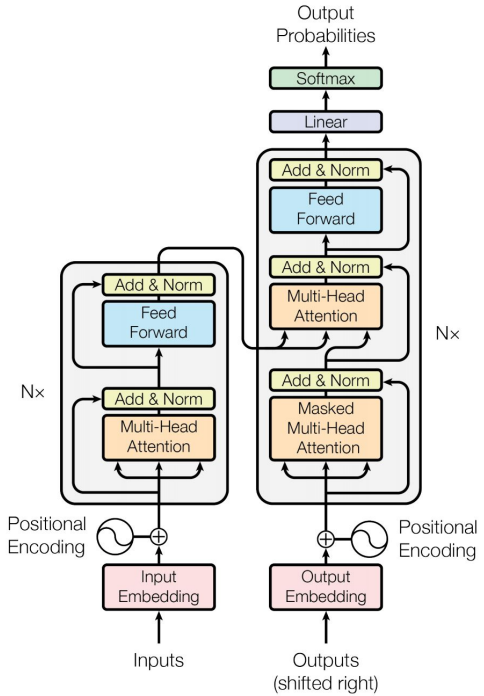
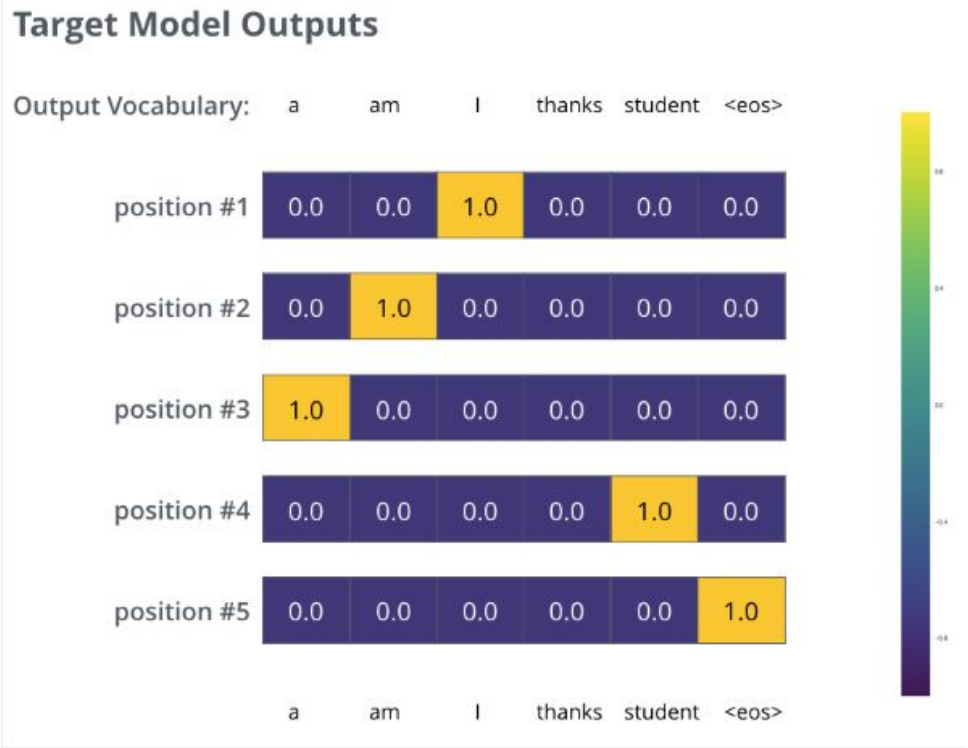


Figure 1: The Transformer - model architecture.

Nous obtenons les sorties à longueur variable en utilisant la recherche en faisceau.



Alors pourquoi utiliser des transformers ?

Bonnes propriétés de calcul

- La quantité de calcul par couche est réduite.
- Le nombre de couches (c'est-à-dire la longueur du chemin le long du graphique de calcul) est réduit.
- Presque tous les calculs sont parallélisables.

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Des performances de pointe

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [15]	23.75			
Deep-Att + PosUnk [32]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [31]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [8]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [26]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [32]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [31]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [8]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.0**	$2.3 \cdot 10^{19}$	

Non, sérieusement. Des performances de pointe.

2. Machine Translation

Research Paper	Datasets	Metric	Source Code	Year
Understanding Back-Translation at Scale	• WMT 2014 English-to-French • WMT 2014 English-to-German	• BLEU: 45.6 • BLEU: 35.0	• PyTorch	2018
WEIGHTED TRANSFORMER NETWORK FOR MACHINE TRANSLATION	• WMT 2014 English-to-French • WMT 2014 English-to-German	• BLEU: 41.4 • BLEU: 28.9	• NOT FOUND	2017
Attention Is All You Need	• WMT 2014 English-to-French • WMT 2014 English-to-German	• BLEU: 41.0 • BLEU: 28.4	• PyTorch • Tensorflow	2017
NON-AUTOREGRESSIVE NEURAL MACHINE TRANSLATION	• WMT16 Ro→En	• BLEU: 31.44	• PyTorch	2017
Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets	• NIST02 • NIST03 • NIST04 • NIST05	• 38.74 • 36.01 • 37.54 • 33.76	• NMTPY	2017

Transformers en PyTorch et TensorFlow

Attention RNN : le codeur

```python
class Encoder(nn.Module):
    ''' A encoder model with self attention mechanism. '''

    def __init__(
            self,
            n_src_vocab, len_max_seq, d_word_vec,
            n_layers, n_head, d_k, d_v,
            d_model, d_inner, dropout=0.1):

        super().__init__()

        n_position = len_max_seq + 1

        self.src_word_emb = nn.Embedding(
            n_src_vocab, d_word_vec, padding_idx=Constants.PAD)

        self.position_enc = nn.Embedding.from_pretrained(
            get_sinusoid_encoding_table(n_position, d_word_vec, padding_idx=0),
            freeze=True)

        self.layer_stack = nn.ModuleList([
            EncoderLayer(d_model, d_inner, n_head, d_k, d_v, dropout=dropout)
            for _ in range(n_layers)])
```

```python
def forward(self, src_seq, src_pos, return_attns=False):

    enc_slf_attn_list = []

    # -- Prepare masks
    slf_attn_mask = get_attn_key_pad_mask(seq_k=src_seq, seq_q=src_seq)
    non_pad_mask = get_non_pad_mask(src_seq)

    # -- Forward
    enc_output = self.src_word_emb(src_seq) + self.position_enc(src_pos)

    for enc_layer in self.layer_stack:
        enc_output, enc_slf_attn = enc_layer(
            enc_output,
            non_pad_mask=non_pad_mask,
            slf_attn_mask=slf_attn_mask)
        if return_attns:
            enc_slf_attn_list += [enc_slf_attn]

    if return_attns:
        return enc_output, enc_slf_attn_list
    return enc_output,
```

Attention RNN : le décodeur

```python
class Decoder(nn.Module):
    ''' A decoder model with self attention mechanism. '''

    def __init__(
            self,
            n_tgt_vocab, len_max_seq, d_word_vec,
            n_layers, n_head, d_k, d_v,
            d_model, d_inner, dropout=0.1):

        super().__init__()
        n_position = len_max_seq + 1

        self.tgt_word_emb = nn.Embedding(
            n_tgt_vocab, d_word_vec, padding_idx=Constants.PAD)

        self.position_enc = nn.Embedding.from_pretrained(
            get_sinusoid_encoding_table(n_position, d_word_vec, padding_idx=0),
            freeze=True)

        self.layer_stack = nn.ModuleList([
            DecoderLayer(d_model, d_inner, n_head, d_k, d_v, dropout=dropout)
            for _ in range(n_layers)])
```

```python
def forward(self, tgt_seq, tgt_pos, src_seq, enc_output, return_attns=False):

    dec_slf_attn_list, dec_enc_attn_list = [], []

    # -- Prepare masks
    non_pad_mask = get_non_pad_mask(tgt_seq)

    slf_attn_mask_subseq = get_subsequent_mask(tgt_seq)
    slf_attn_mask_keypad = get_attn_key_pad_mask(seq_k=tgt_seq, seq_q=tgt_seq)
    slf_attn_mask = (slf_attn_mask_keypad + slf_attn_mask_subseq).gt(0)

    dec_enc_attn_mask = get_attn_key_pad_mask(seq_k=src_seq, seq_q=tgt_seq)

    # -- Forward
    dec_output = self.tgt_word_emb(tgt_seq) + self.position_enc(tgt_pos)

    for dec_layer in self.layer_stack:
        dec_output, dec_slf_attn, dec_enc_attn = dec_layer(
            dec_output, enc_output,
            non_pad_mask=non_pad_mask,
            slf_attn_mask=slf_attn_mask,
            dec_enc_attn_mask=dec_enc_attn_mask)

        if return_attns:
            dec_slf_attn_list += [dec_slf_attn]
            dec_enc_attn_list += [dec_enc_attn]

    if return_attns:
        return dec_output, dec_slf_attn_list, dec_enc_attn_list
    return dec_output,
```

La librairie transformers - python

Transformers

Architectures :

BERT (from Google)
GPT (from OpenAI)
GPT-2 (from OpenAI)
Transformer-XL (from Google/CMU)
XLNet (from Google/CMU)
XLM (from Facebook)

RoBERTa (from Facebook)
DistilBERT (from HuggingFace)
CTRL (from Salesforce)
CamemBERT (from Inria/Facebook/Sorbonne)
ALBERT (from Google Research and the Toyota Technological Institute at Chicago)
T5 (from Google AI)
XLM-RoBERTa (from Facebook AI)
MMBT (from Facebook)

FlauBERT (from CNRS)

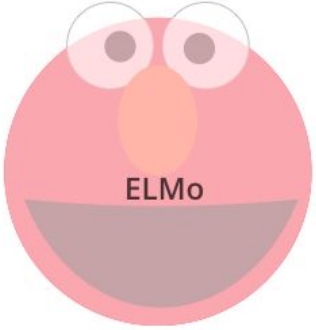
Modèles pré-entraînés notables


THE
TRANSFORMER


ULM-FiT


BERT

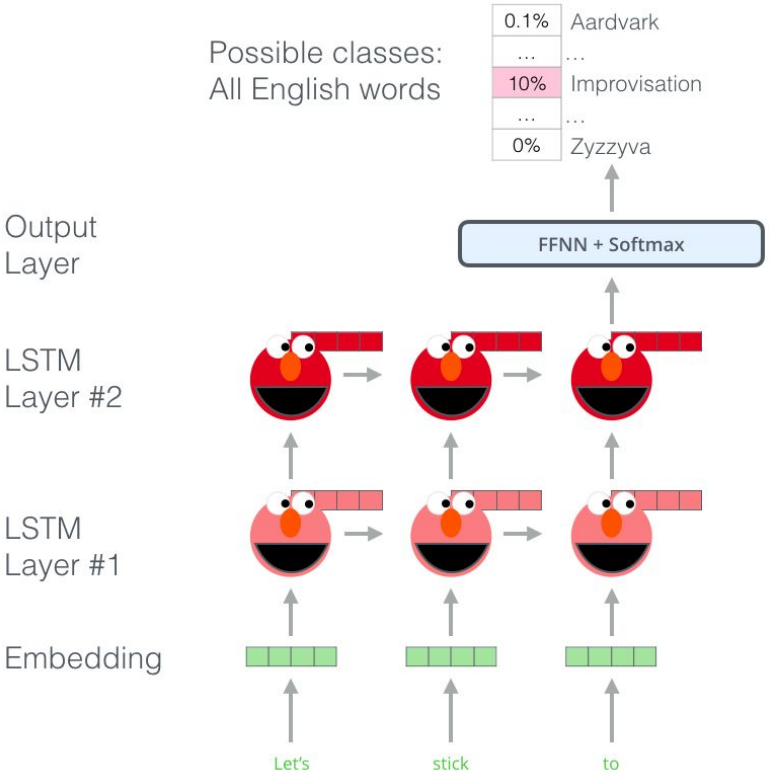

OpenAI
Transformer


ELMo

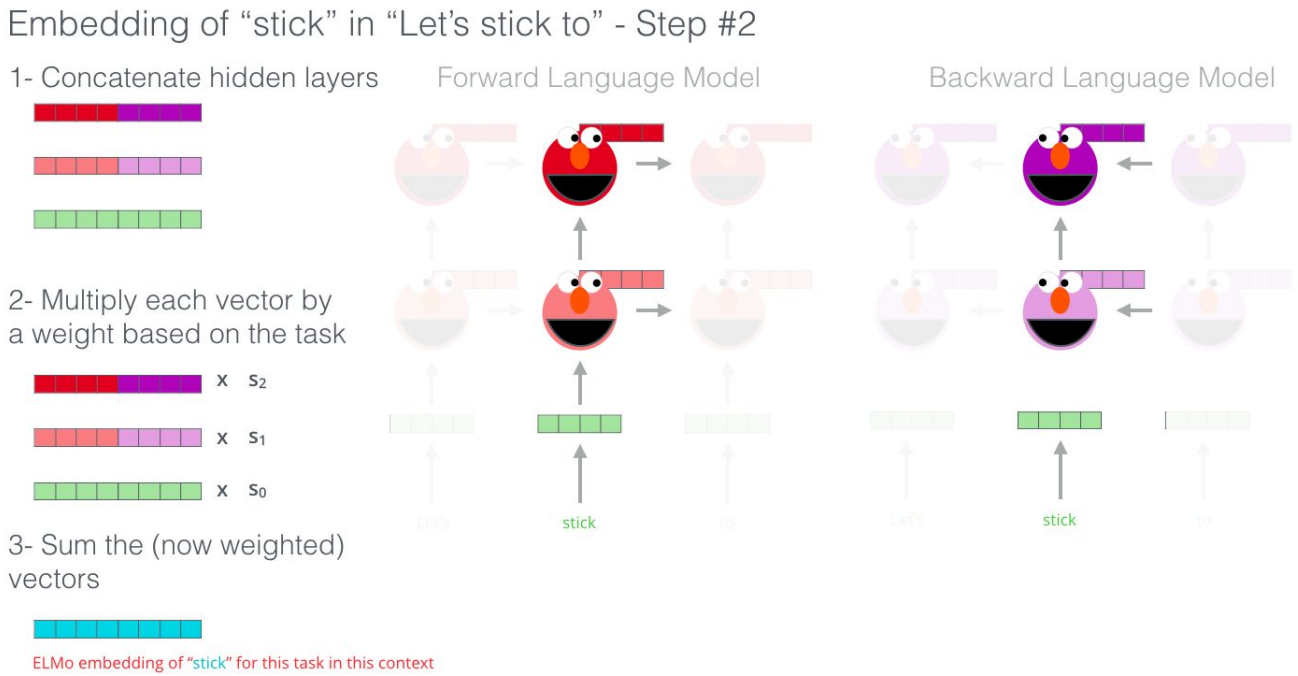
ELMo : intégration des modèles linguistiques
BiLSTM préformé pour l'intégration contextuelle

La désambiguïsation basée sur le contexte est difficile.

Le biLSTM d'ELMo est pré-entraîné sur un modèle linguistique.



L'intégration d'un mot donné dans la phrase par ELMo est la concaténation des états cachés de son biLSTM pour le mot.



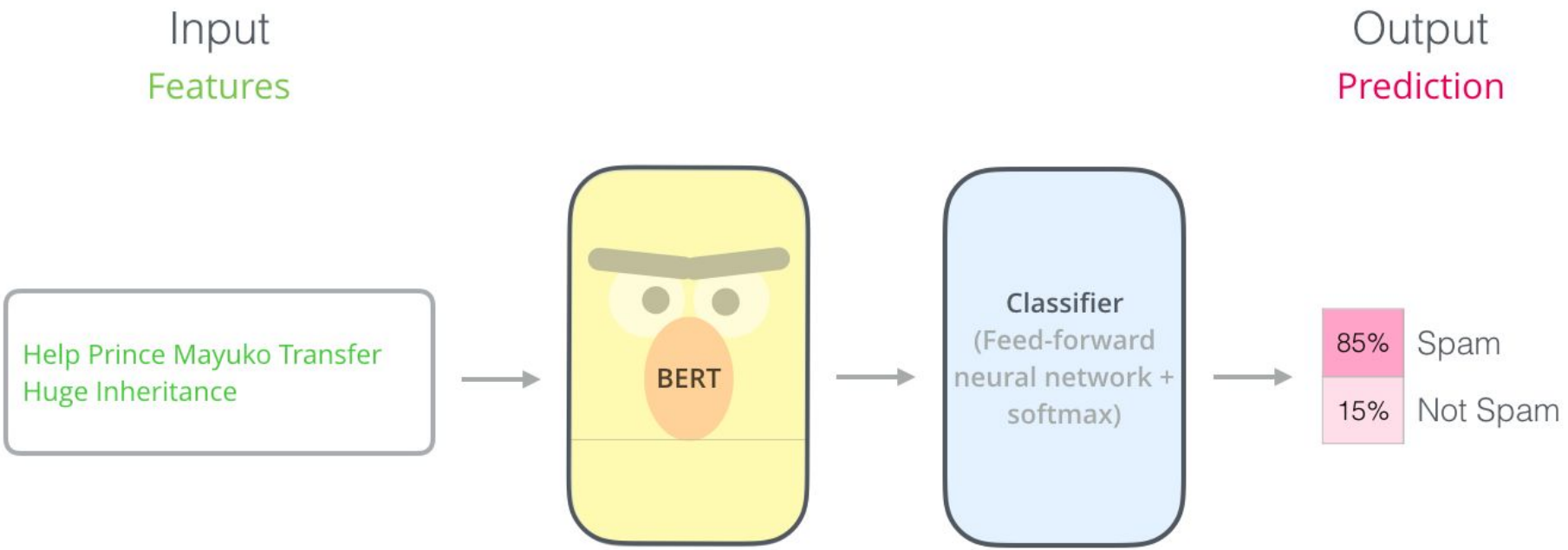
ELMo : biLSTM pour l'intégration de mots neuronaux

ELMo representations are:

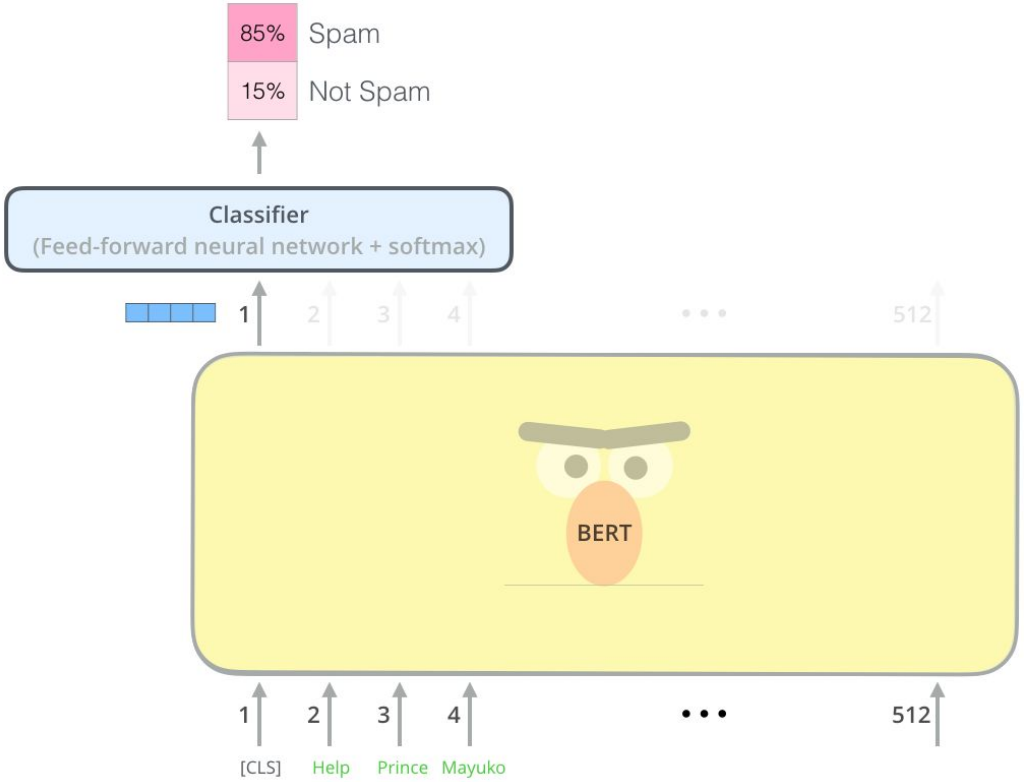
- *Contextual*: The representation for each word depends on the entire context in which it is used.
- *Deep*: The word representations combine all layers of a deep pre-trained neural network.
- *Character based*: ELMo representations are purely character based, allowing the network to use morphological clues to form robust representations for out-of-vocabulary tokens unseen in training.

BERT : Représentations de codeurs bidirectionnels à partir de transformers
Codeur à transformer pré-entraîné pour l'intégration des phrases

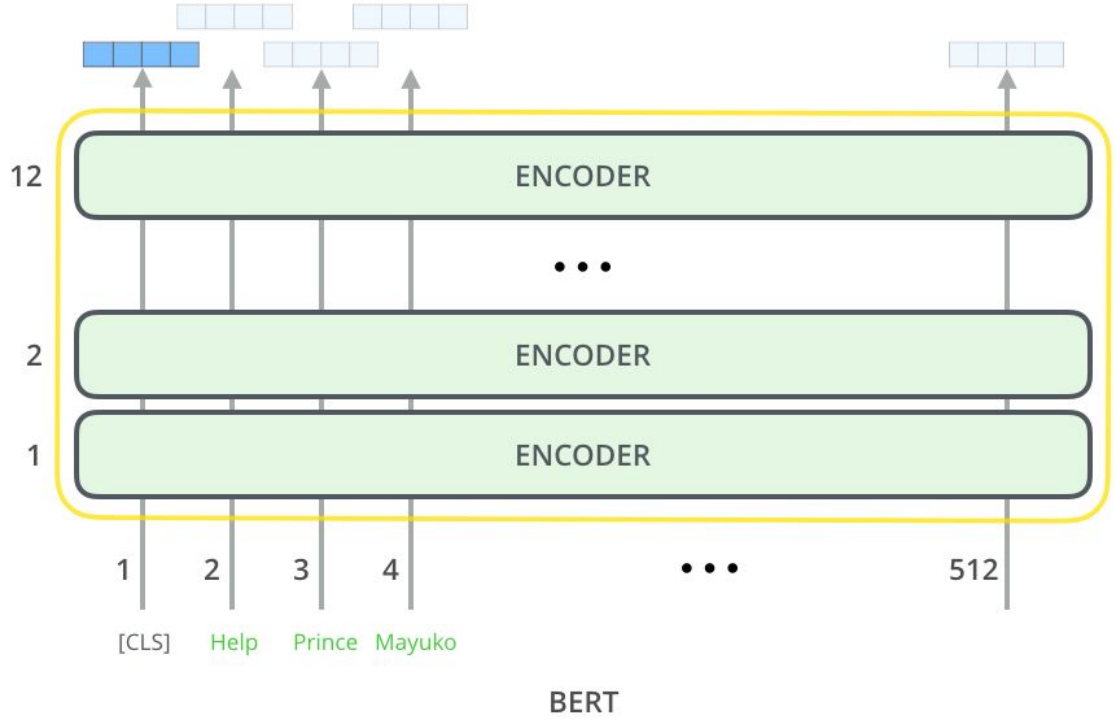
BERT est ImageNet pour les langues



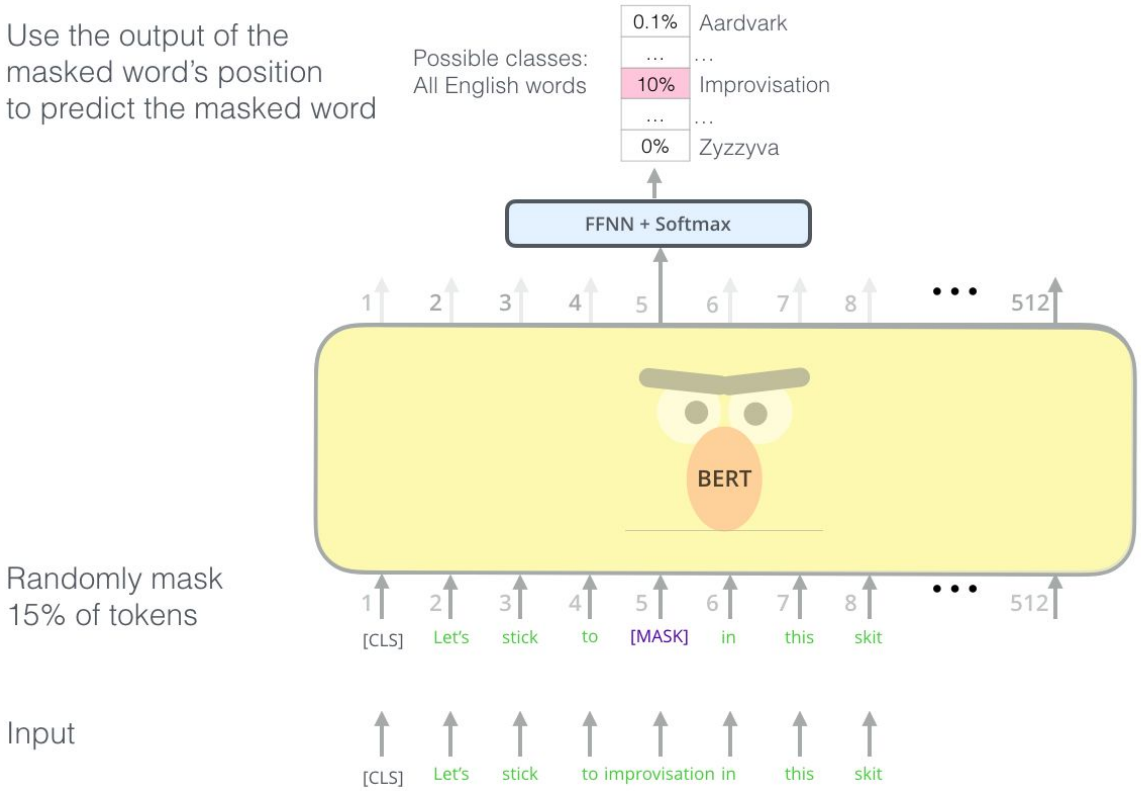
BERT est ImageNet pour les langues



L'architecture du BERT n'est qu'une pile de codeurs d'un transformer.



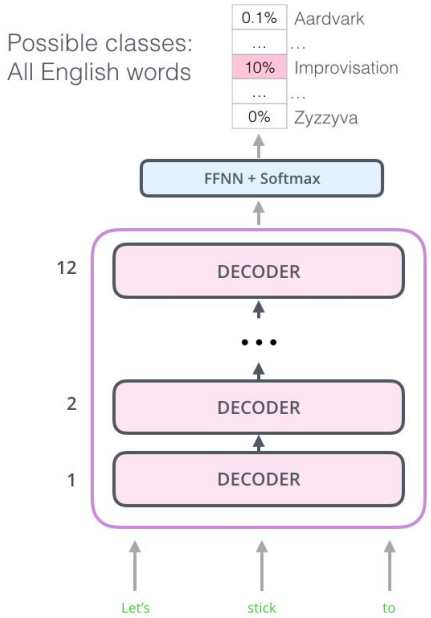
BERT est formé comme un modèle skip-gram



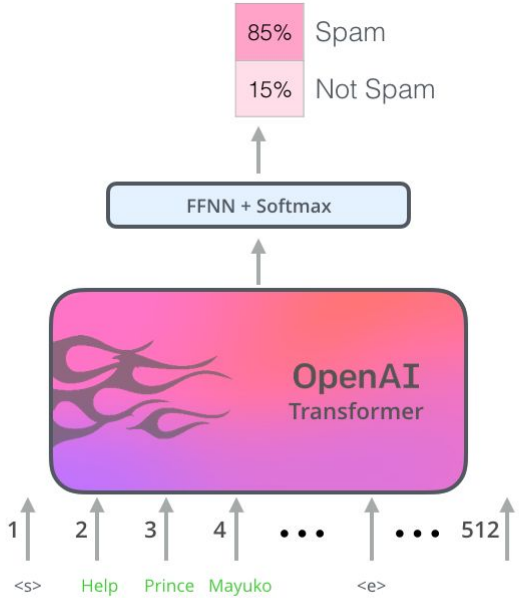
"On ne peut pas condenser la signification d'une phrase entière de %&!$# dans un seul vecteur $&!#* !"
-- Ray Mooney, Association pour la linguistique informatique (ACL) 2014

Le transformateur d'OpenAI
Décodeur transformateur pré-entraîné pour la modélisation du langage

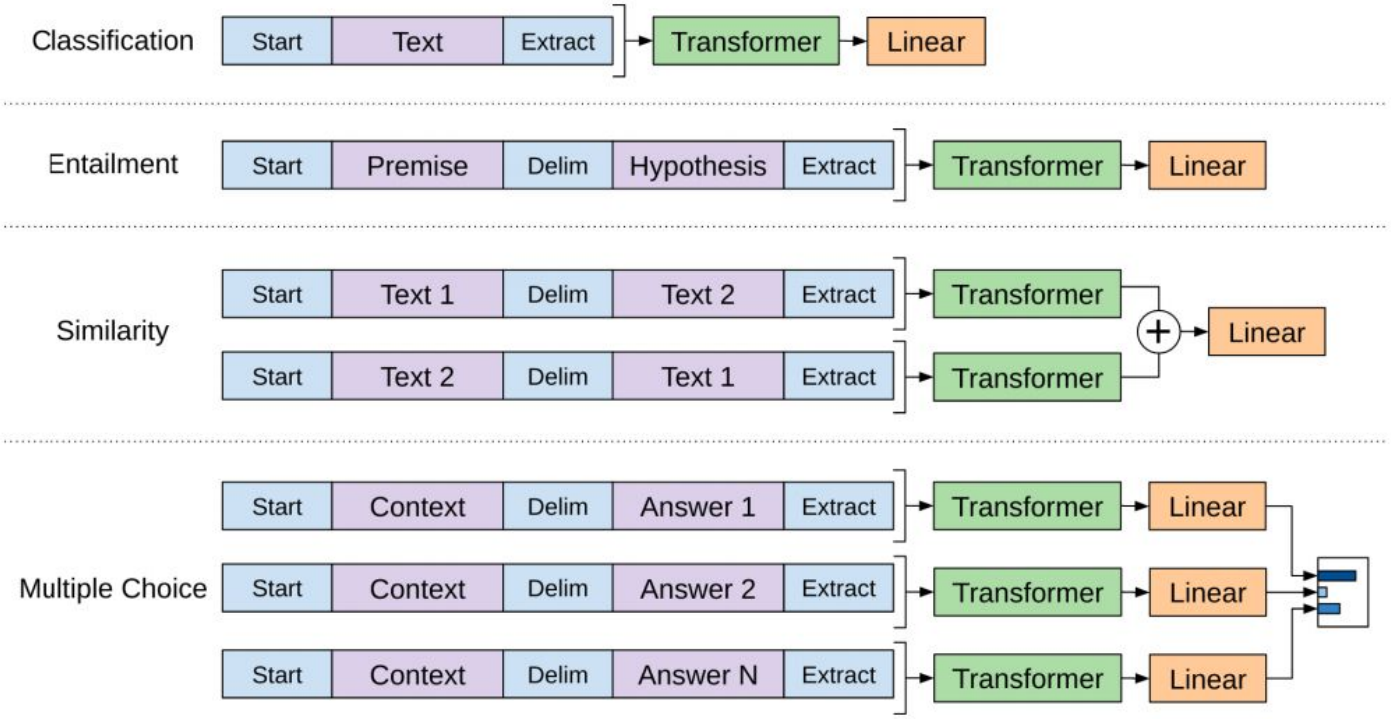
Le transformateur OpenAI original n'est qu'une pile de décodeurs formés à la modélisation du langage (sans supervision).



Comme pour le BERT, vous pouvez utiliser le modèle préformé pour n'importe quelle tâche.



Différentes tâches utilisent le transformer OpenAI de différentes manières.

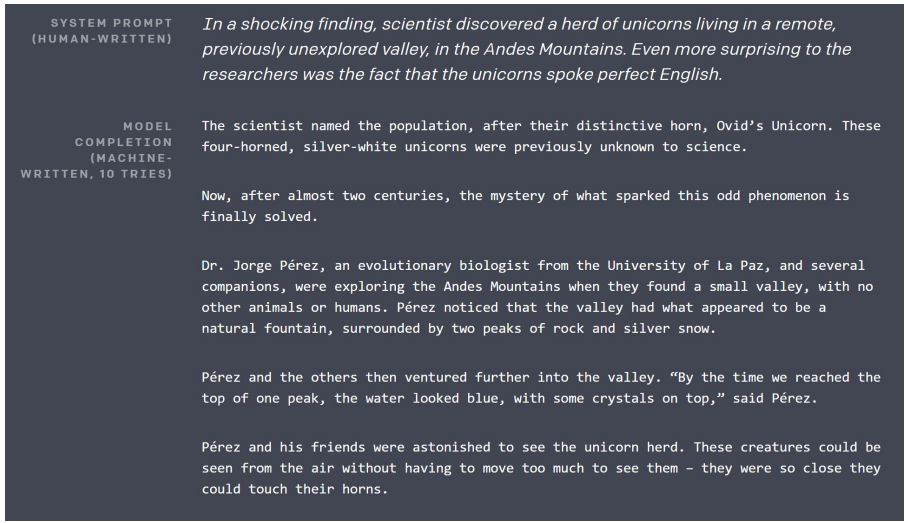


OpenAI a testé son transformer sur l'apprentissage à froid (zero shot learning)

"Pour SST-2 (analyse des sentiments), nous ajoutons un jeton à chaque exemple et limitons la distribution de sortie du modèle de langage aux seuls mots positifs et négatifs et le jeton auquel il attribue la plus grande probabilité comme prédiction"

Le GPT-2 d'Open AI n'est qu'un très, très gros transformer.

- 1,5 milliard de paramètres !
- Formé sur 8 millions de pages web !
  - a supprimé tous les liens sortants sur Reddit avec au moins trois votes positifs.



SYSTEM PROMPT (HUMAN-WRITTEN)	*In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.*
MODEL COMPLETION (MACHINE-WRITTEN, 10 TRIES)	The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science. Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved. Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow. Pérez and the others then ventured further into the valley. "By the time we reached the top of one peak, the water looked blue, with some crystals on top," said Pérez. Pérez and his friends were astonished to see the unicorn herd. These creatures could be seen from the air without having to move too much to see them – they were so close they could touch their horns.