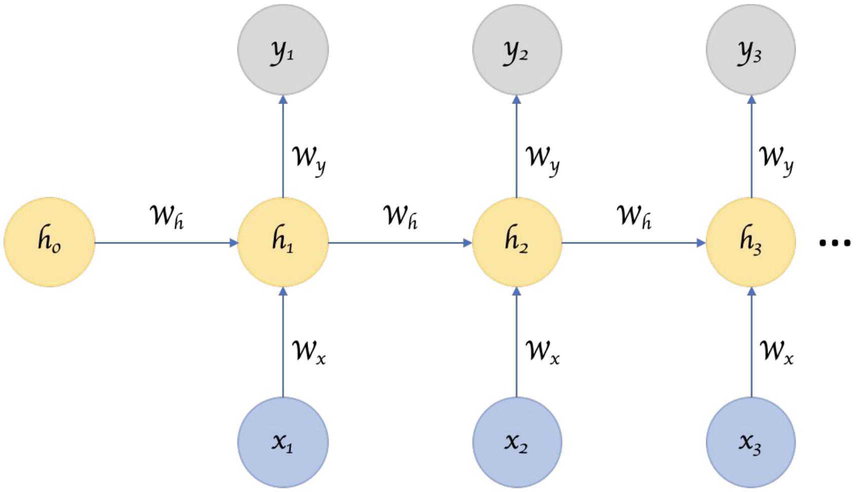
Deep Learning

Lecture 5

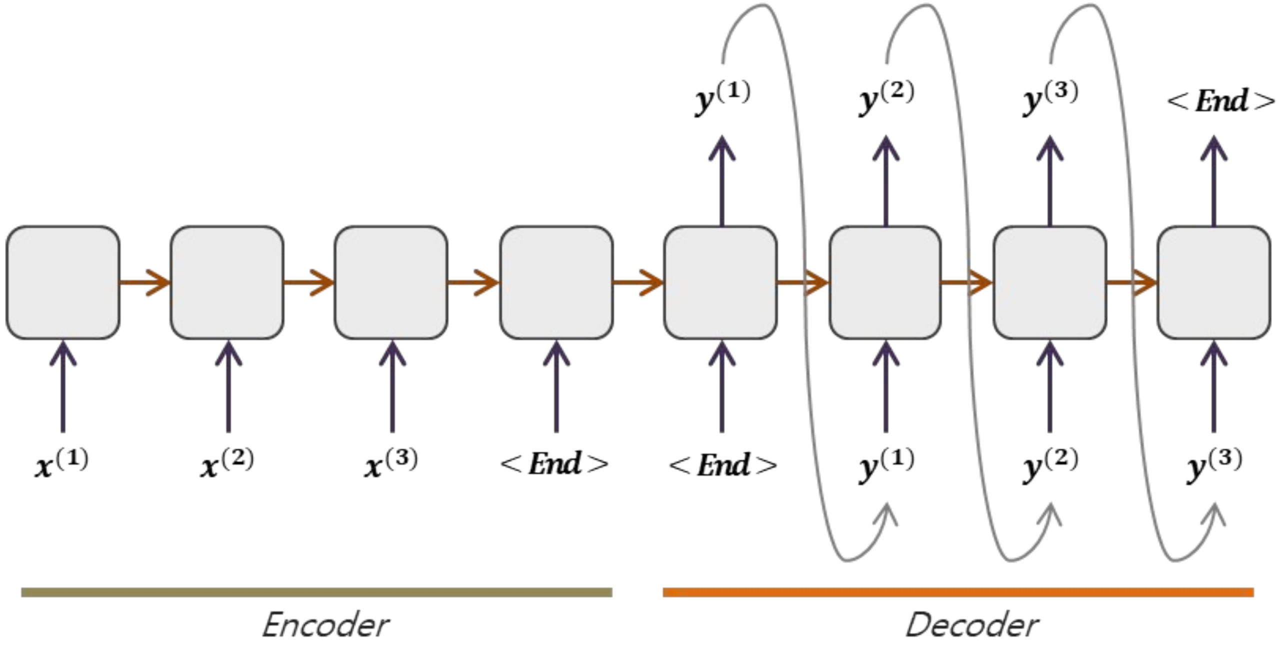
Recap



- RNN
  - Backward
  - Vanishing gradients
  - Exploding gradients
- LSTM
- GRU
- Applications
- Dropout and BN

Problem: copy of sequence

aeebdhf# → aeebdhfaeebdhf (aeebdhf *2)



Attention mechanism

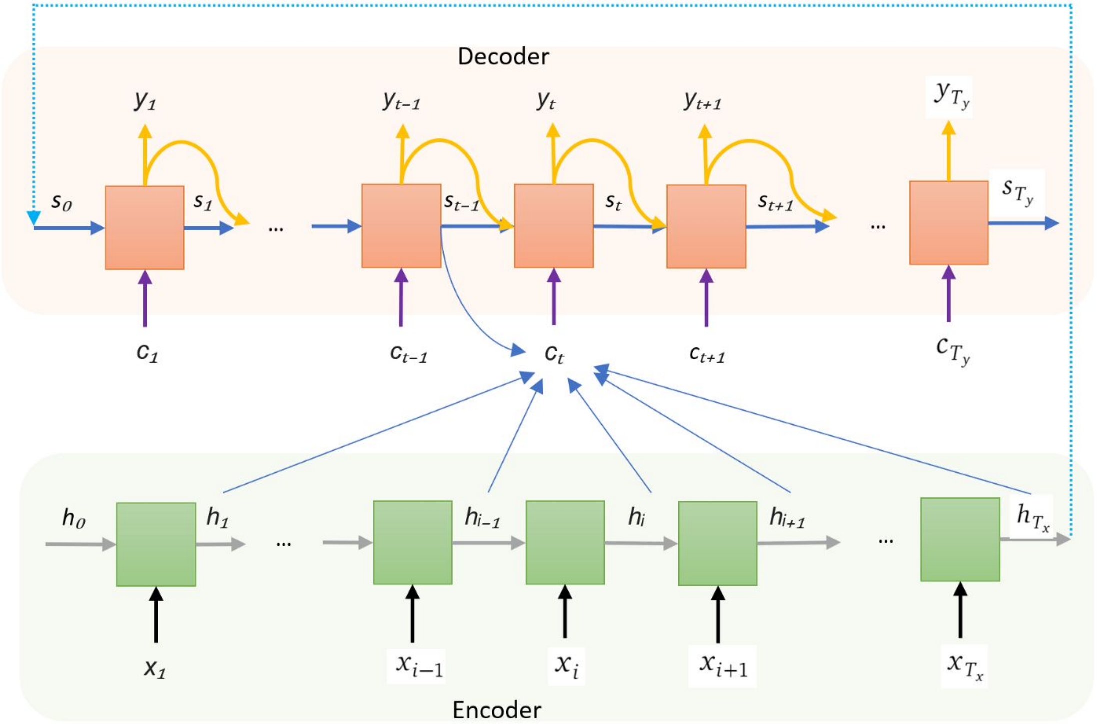
$$s_t \quad h_i$$

$$sim(s_t, h_i) \in \mathbb{R}$$

$$\alpha_{t,j} = \frac{\exp(sim(s_{t-1}, h_j))}{\sum_{i=1}^{T_x} \exp(sim(s_{t-1}, h_i))}$$

$$c_t = \sum_{i=1}^{T_x} \alpha_{t,i} h_i$$

Attention mechanism



$$\alpha_{t,j} = \frac{\exp(sim(s_{t-1}, h_j))}{\sum_{i=1}^{T_x} \exp(sim(s_{t-1}, h_i))}$$

$$c_t = \sum_{i=1}^{T_x} \alpha_{t,i} h_i$$

Similarity function

$$sim(h_1, h_2) - ?$$

1) $h_1^T h_2 \qquad dim(h_1) = dim(h_2)$

2) $\dfrac{h_1^T h_2}{\sqrt{dim(h_1)}} \qquad dim(h_1) = dim(h_2)$

Similarity function

$$sim(h_1, h_2) - ?$$

$$h_{1i} \sim \mathcal{N}(0, 1)$$

$$h_{2i} \sim \mathcal{N}(0, 1)$$

$$\mathbb{E}[h_1^T h_2] = 0$$

$$\mathbb{D}[h_1^T h_2] = dim(h_1)$$

Similarity function

$$sim(h_1, h_2) - ?$$

1) $h_1^T h_2 \qquad dim(h_1) = dim(h_2)$

2) $\dfrac{h_1^T h_2}{\sqrt{dim(h_1)}} \qquad dim(h_1) = dim(h_2)$

3) $w_3 \tanh(w_1 h_1 + w_2 h_2) \qquad dim(h_1) \neq dim(h_2)$

Attention

$$c = \text{Attention}(k: \text{"keys"}, q: \text{"query"}, v: \text{"values"})$$

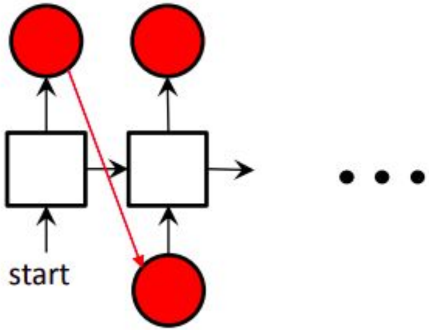
$$\alpha_t = \frac{\exp(sim(q, k_t))}{\sum_{i=1}^{T} \exp(sim(q, k_i))}$$

$$c = \sum_{i=1}^{T} \alpha_i v_i$$

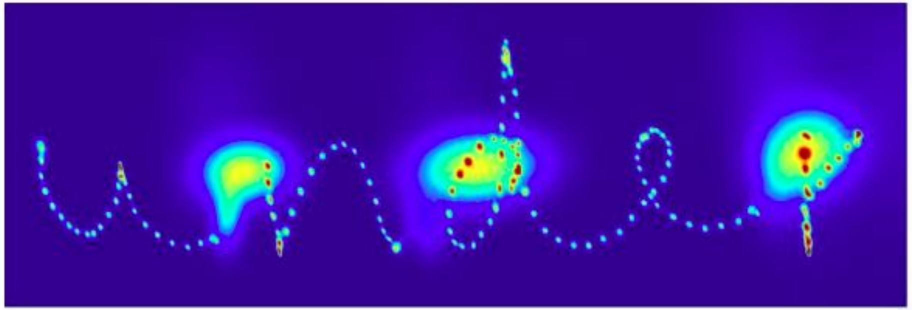
Handwriting generation:

handwriting -> handwriting

Next pen position (we predict parameters):
x1,x2 - mixture of bivariate Gaussians
x3 - Bernoulli distribution

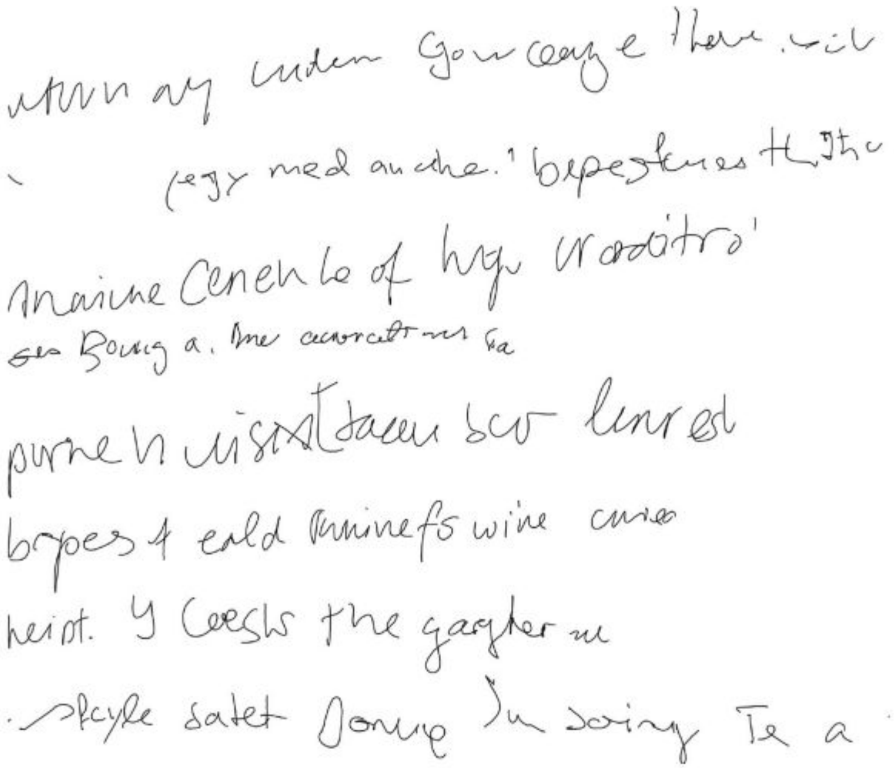


Current pen position:
x1,x2 – pen offset
x3 – is it end of the stroke



Handwriting generation:

example



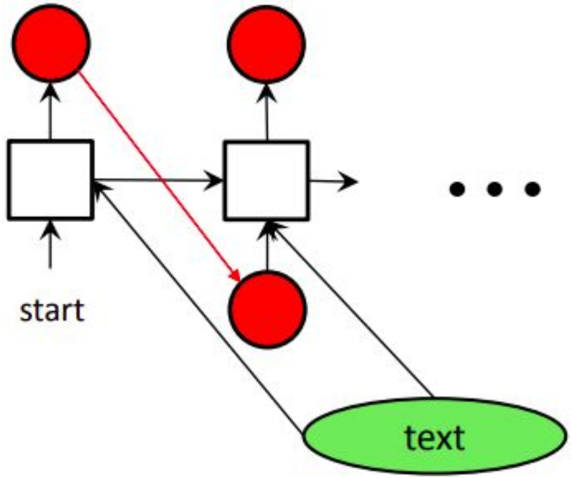
Handwriting synthesis:

text -> handwriting

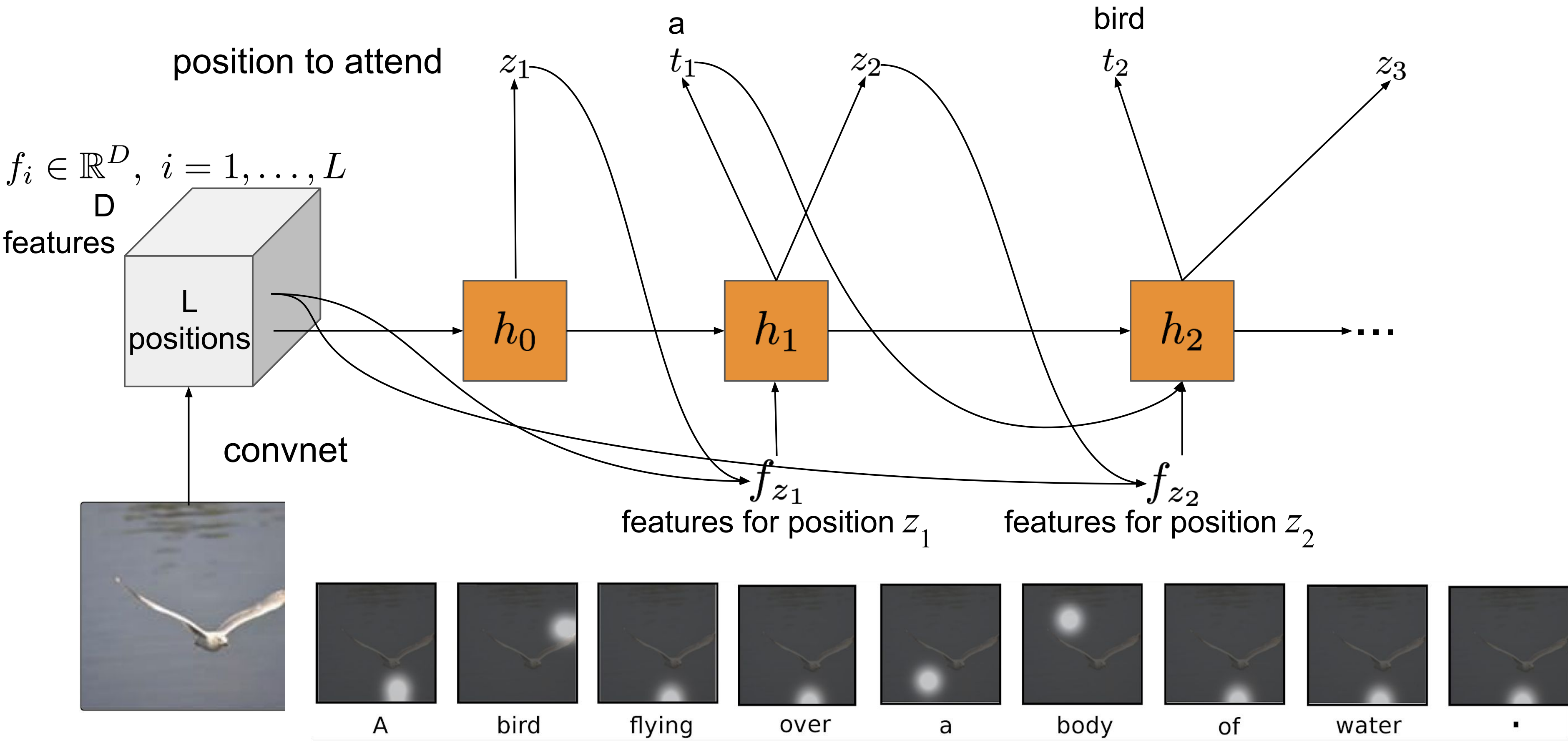
Next pen position

Current pen position

Which letter we write now

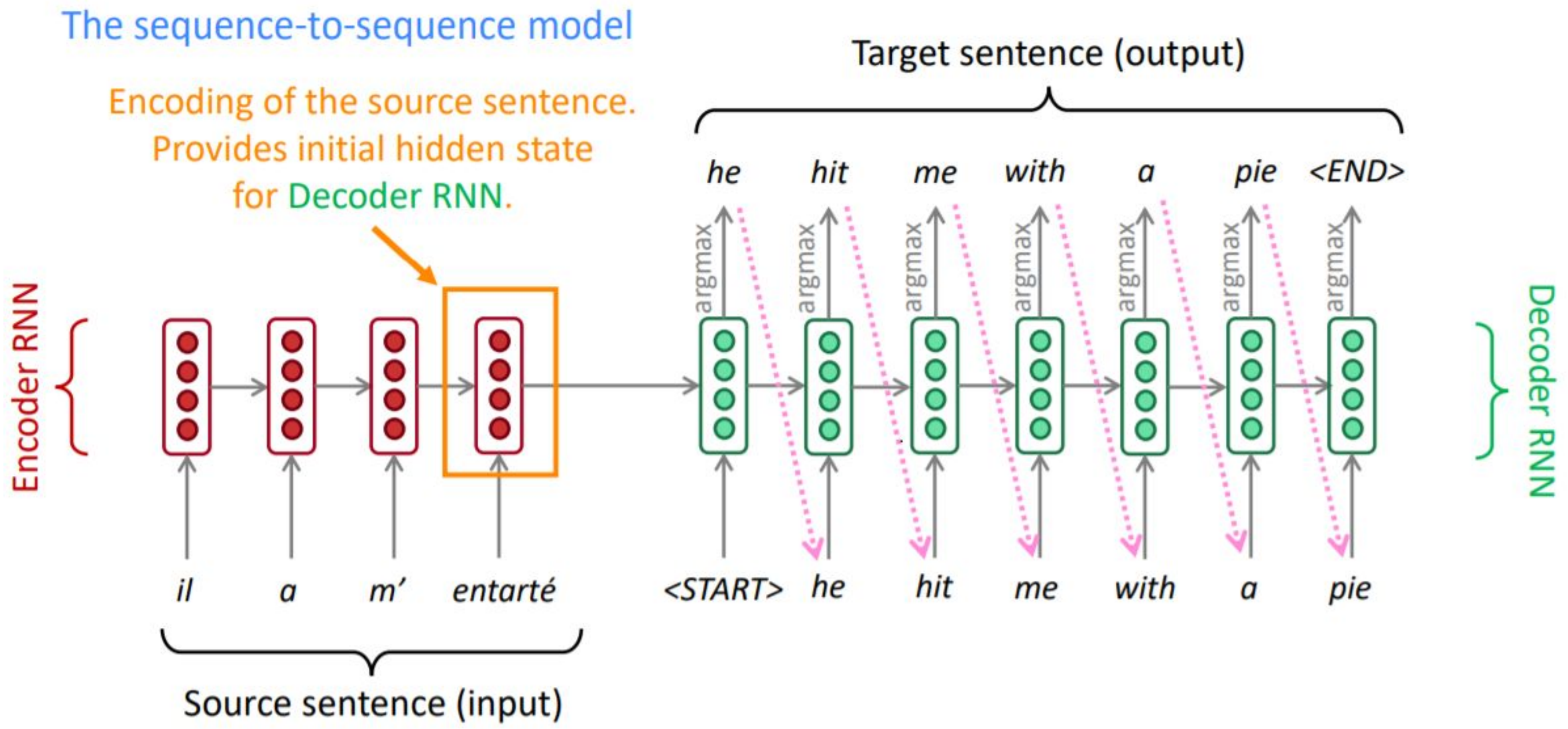


Attention for image captioning



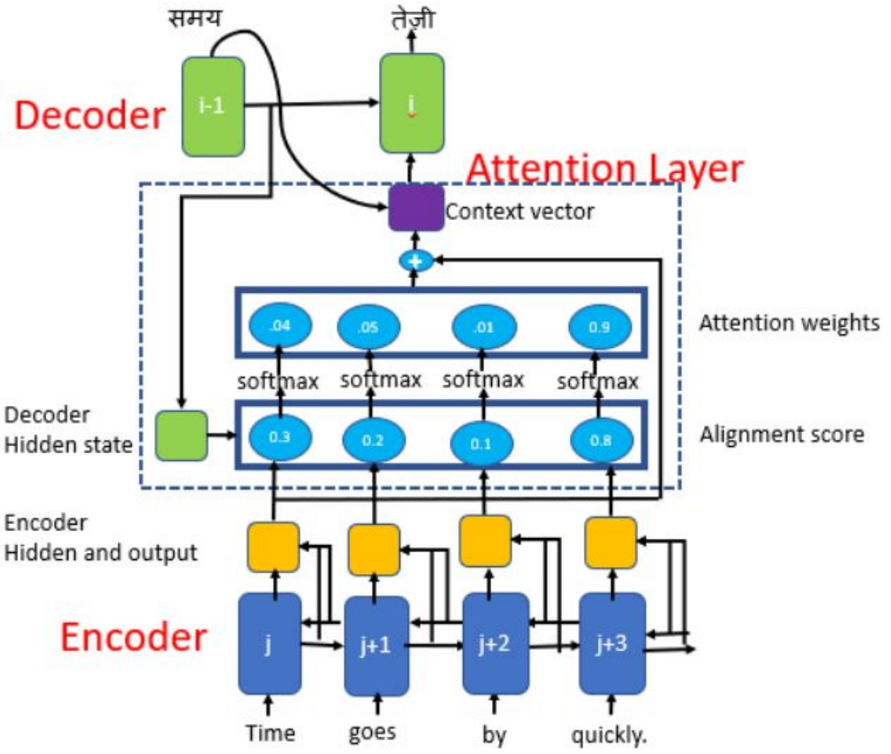
Xu et al. “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention” ICML 2015

Translation with attention

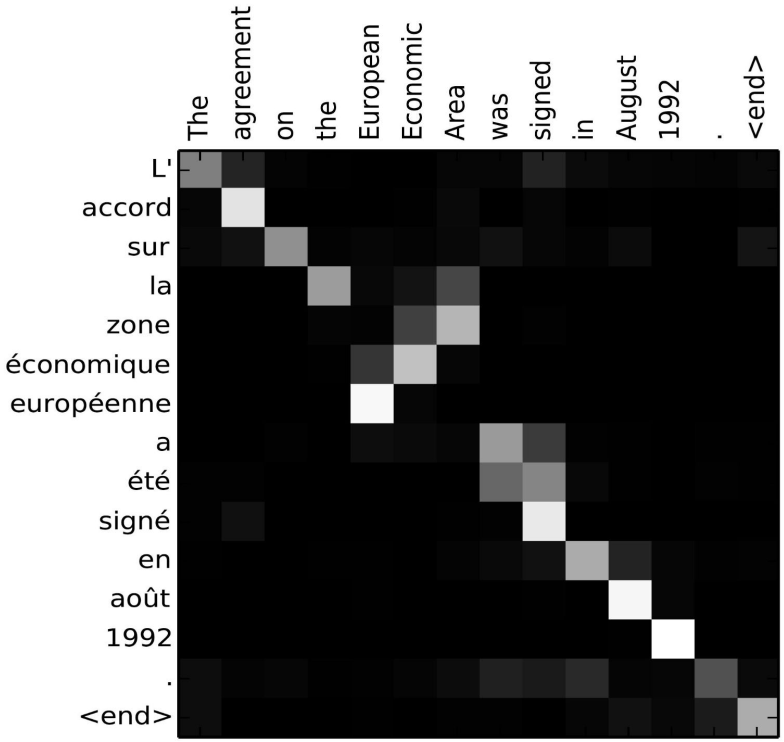


Translation with attention

Translation with attention

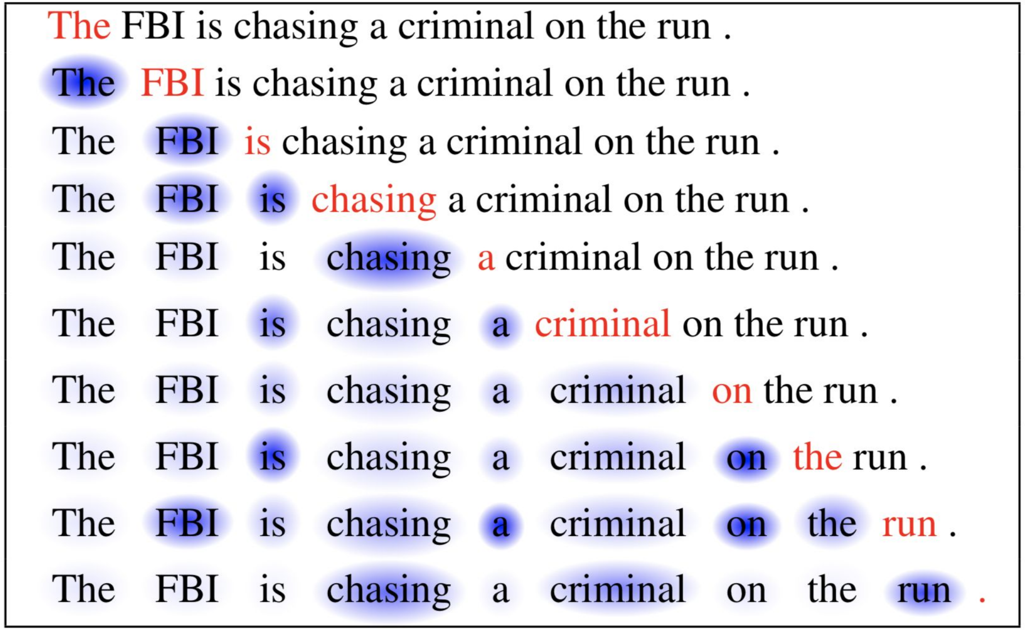


Translation with attention



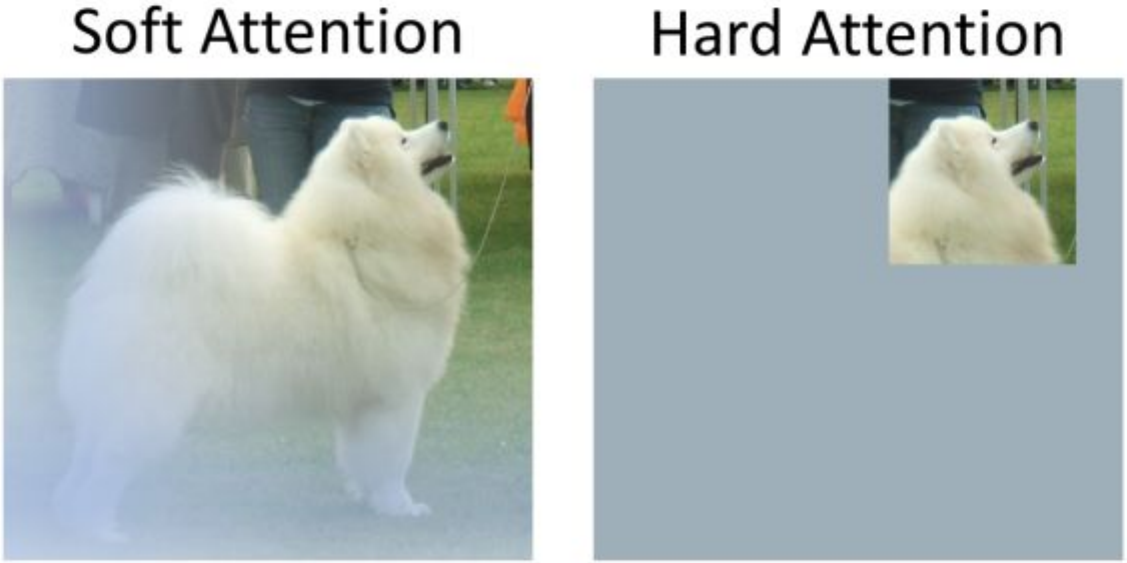
Attention types

Self-attention

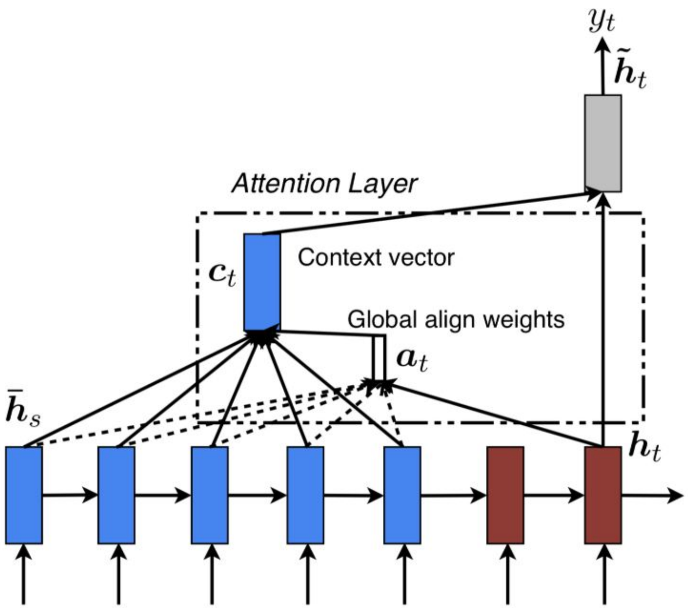


The current word is in red and the size of the blue shade indicates the activation level.

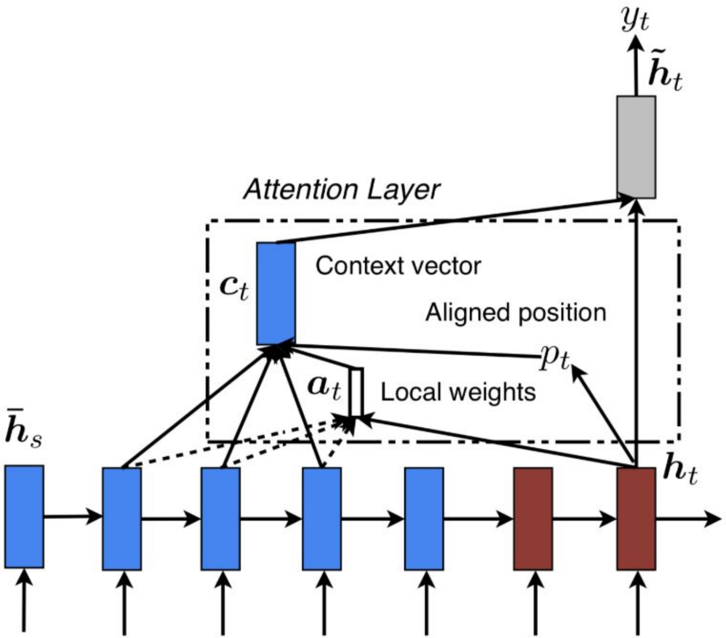
Soft vs Hard Attention



Global vs Local Attention



Global Attention Model



Local Attention Model

What’s next?

Attention is all you need

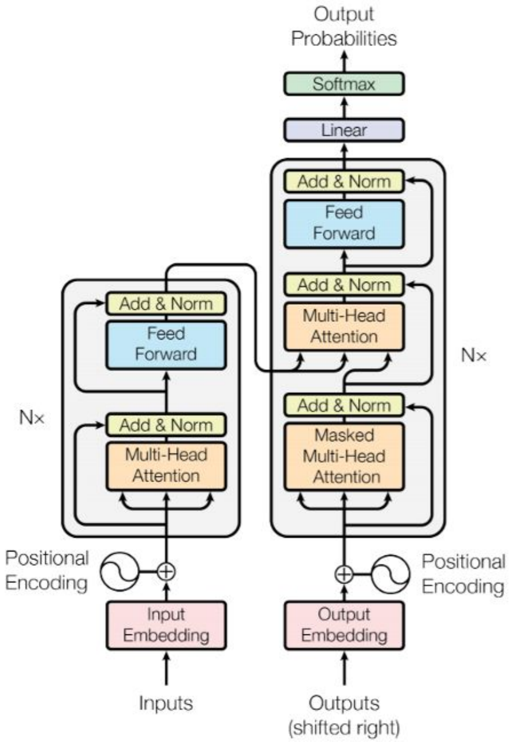
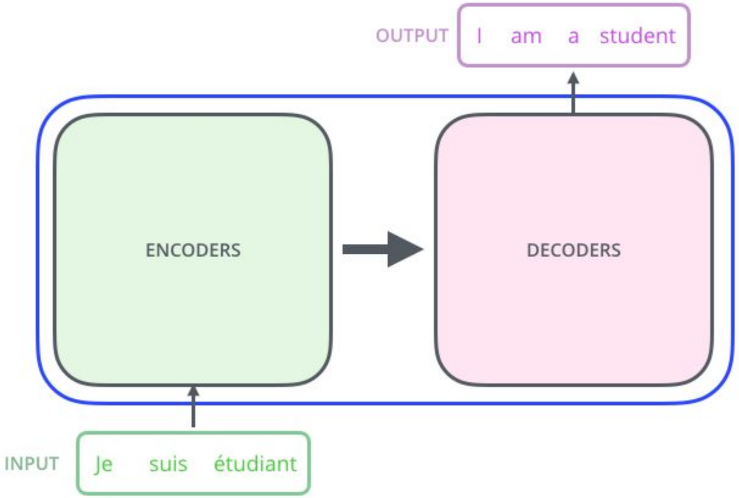
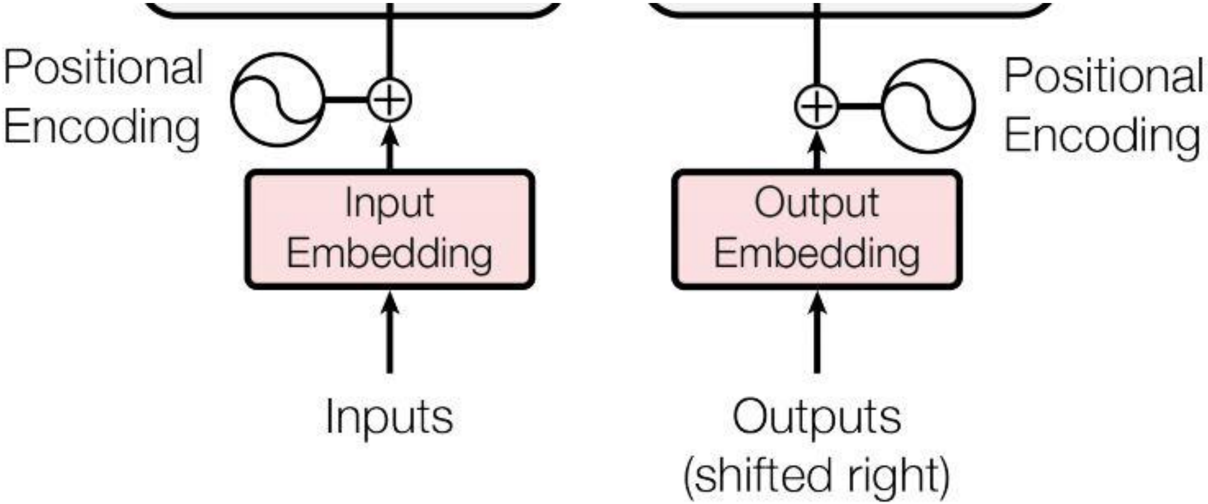


Figure 1: The Transformer - model architecture.



Input layer



Positional
Encoding
Input
Embedding
Inputs
Output
Embedding
Positional
Encoding
Outputs
(shifted right)

Positional encoding

$$PE_{\text{pos},2i} = \sin(\frac{\text{pos}}{10000^{\frac{2i}{d_{model}}}})$$

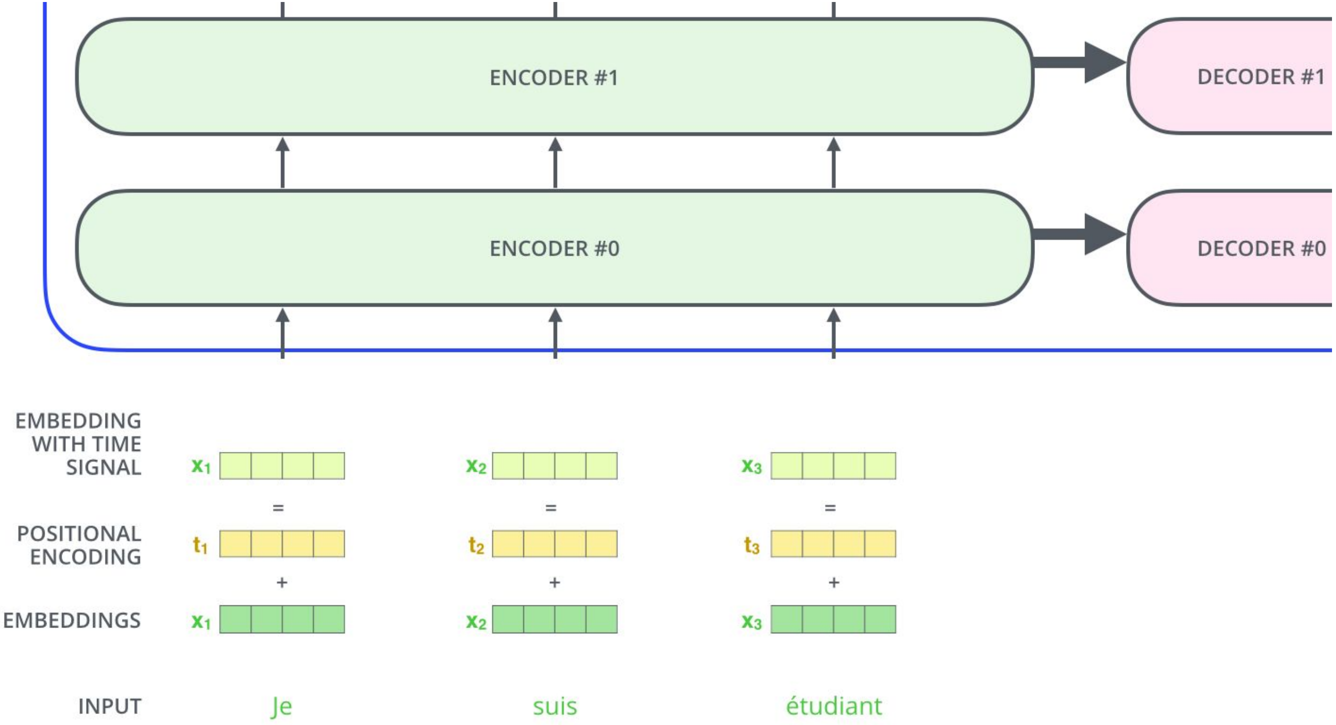
$$PE_{\text{pos},2i+1} = \cos(\frac{\text{pos}}{10000^{\frac{2i}{d_{model}}}})$$

$$PE_{\text{pos}+k,2i} = sin(\frac{\text{pos}+k}{10000^{\frac{2i}{d_{model}}}}) = sin(\frac{\text{pos}}{10000^{\frac{2i}{d_{model}}}} + \frac{k}{10000^{\frac{2i}{d_{model}}}}) =$$

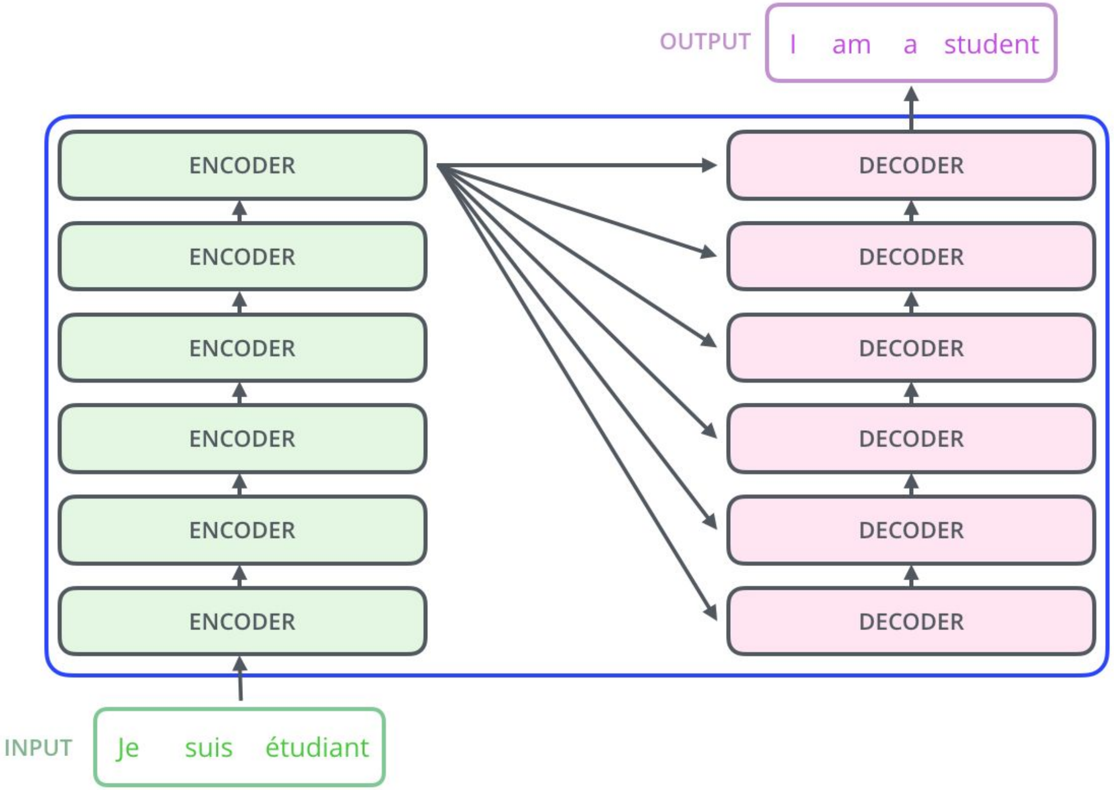
$$= \sin(x+y) = \sin(x)\cos(y) + \cos(x)\sin(y)$$

$$PE_{\text{pos}+k,2i+1} = \cos(x)\cos(y) + \sin(x)\sin(y)$$

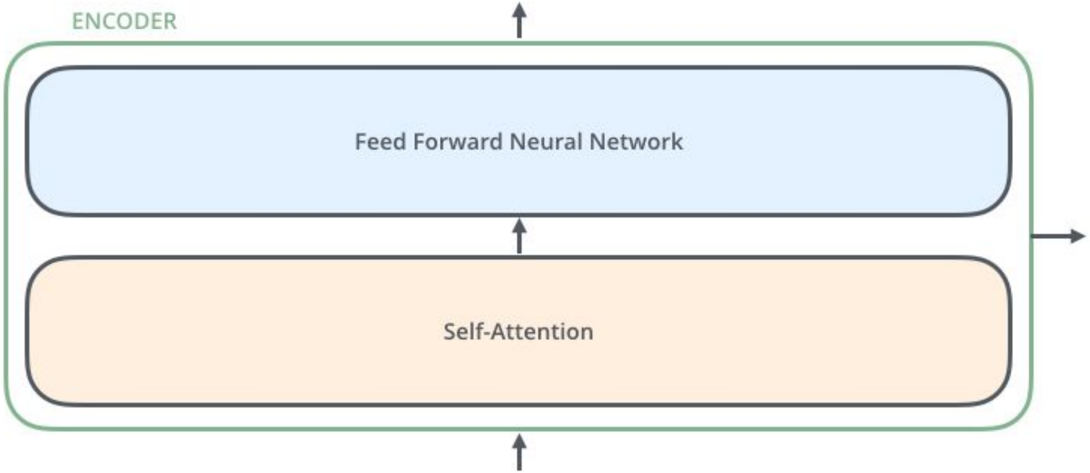
Positional encoding



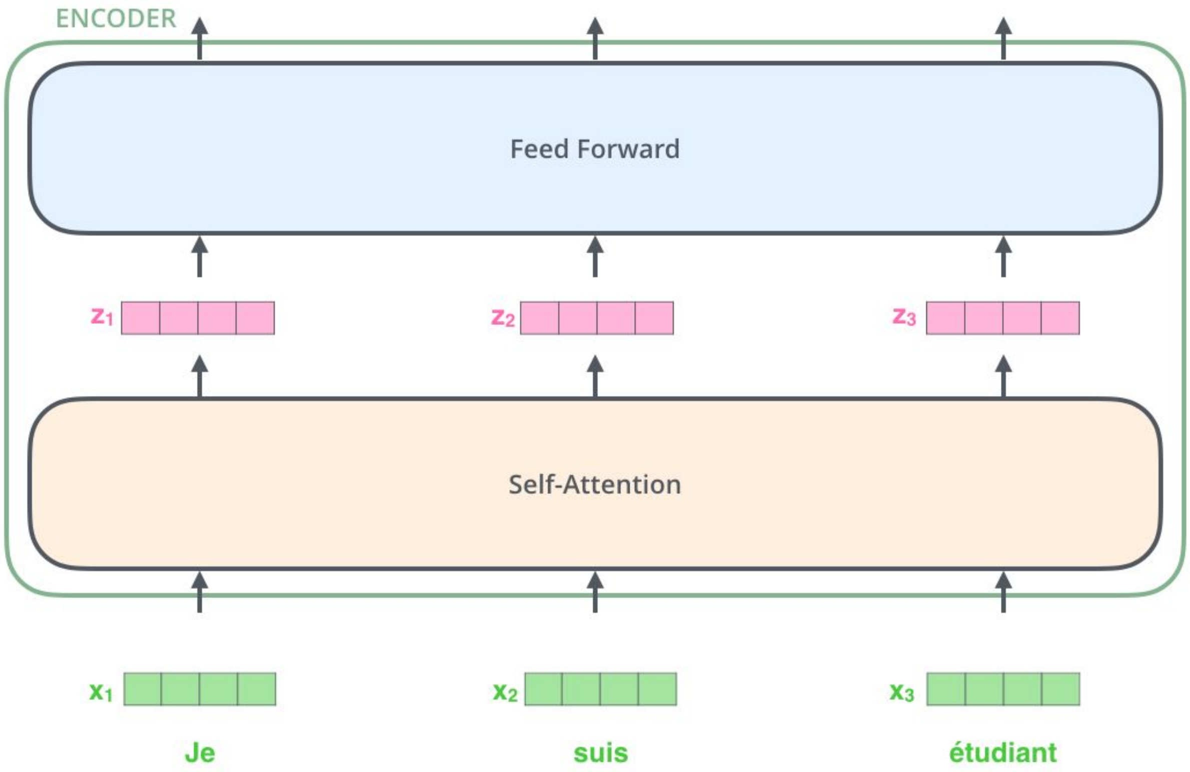
Attention is all you need



Encoder



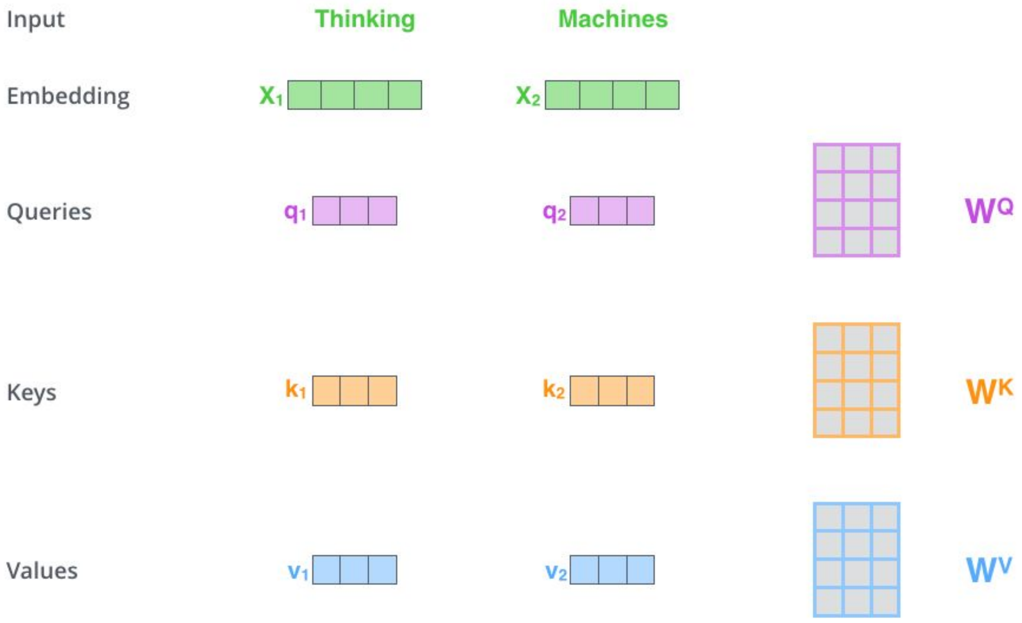
Encoder



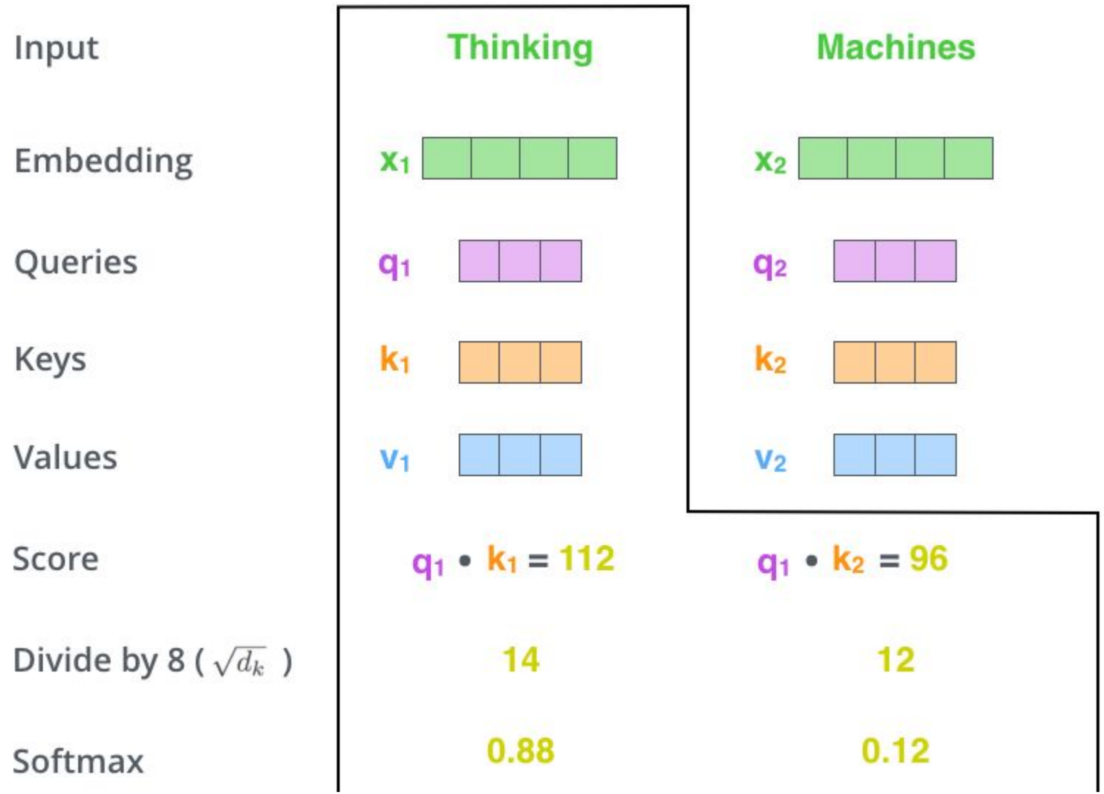
ENCODER
Feed Forward
z1
z2
z3
Self-Attention
x1
x2
x3
Je
suis
étudiant

Self attention

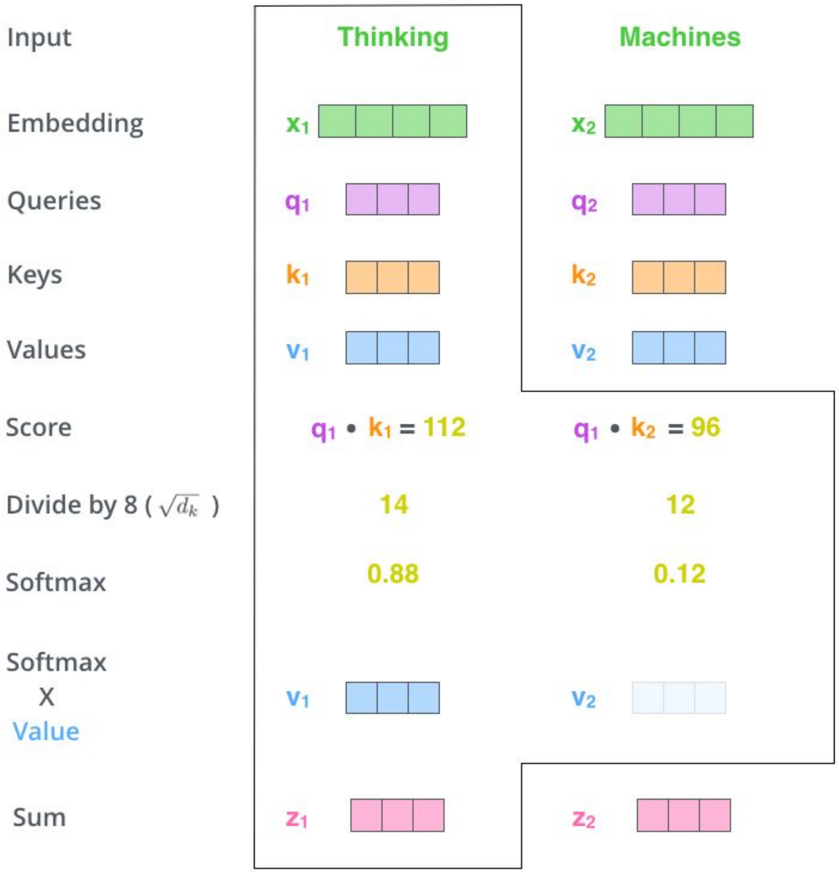
$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{dim}})\mathbf{V}$$



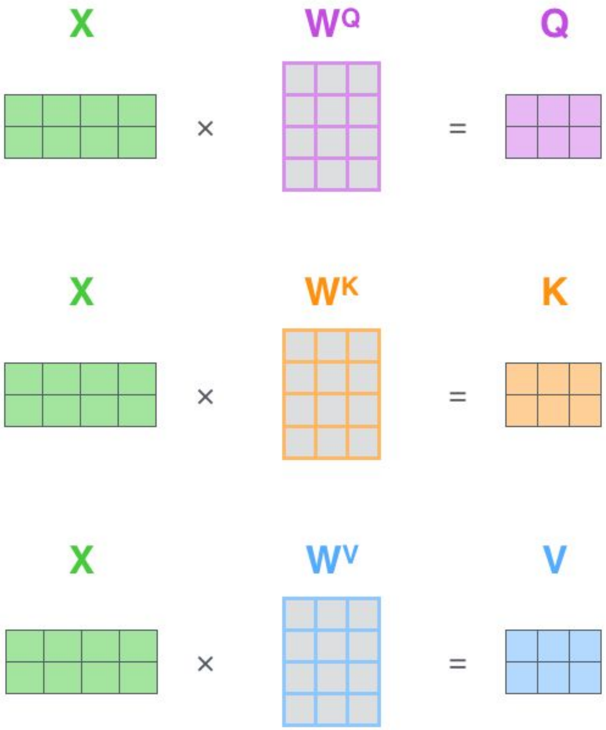
Self attention



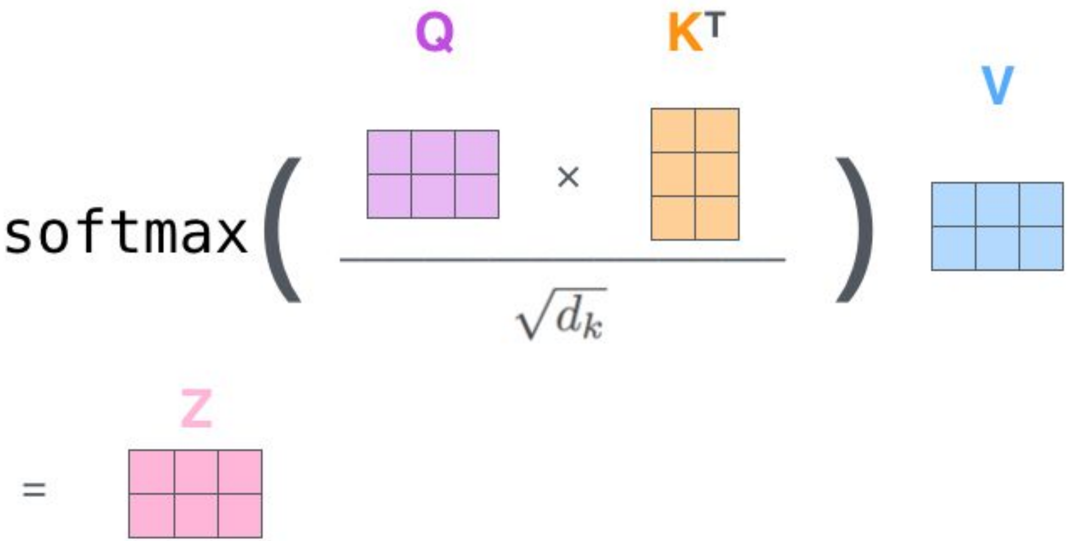
Self attention



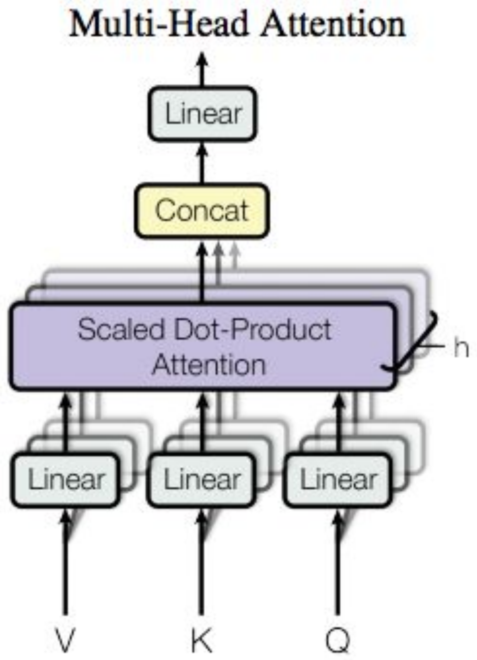
Self attention



Self attention

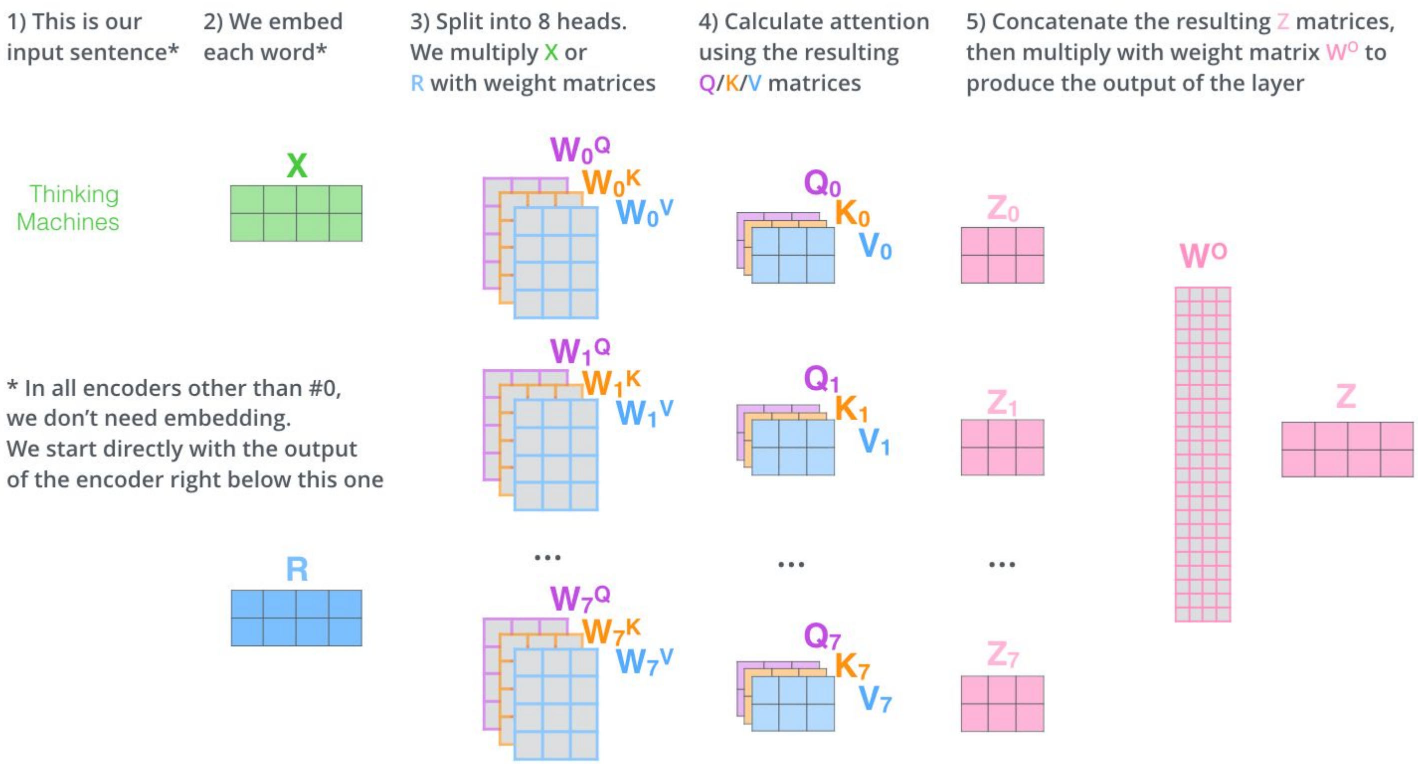


Multi-Head Attention



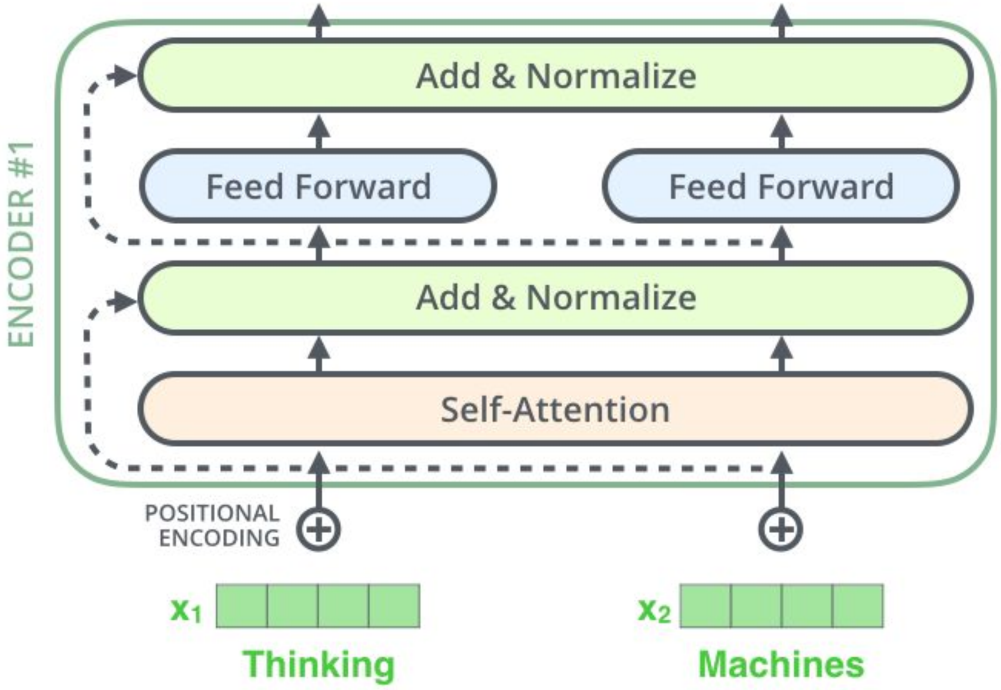
$$\mathrm{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = [\mathrm{head}_1; \ldots; \mathrm{head}_h]\mathbf{W}^O$$
$$\mathrm{head}_i = \mathrm{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V)$$

Multi-Head Attention

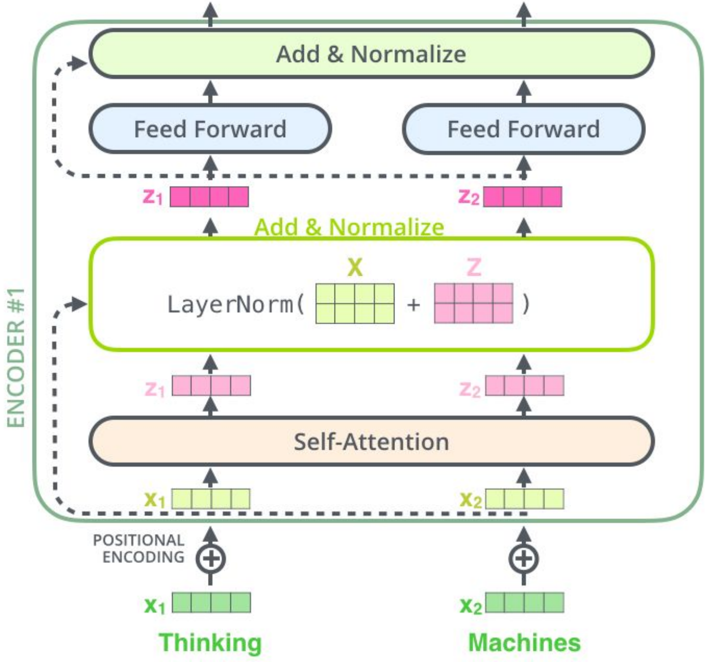


https://jalammar.github.io/illustrated-transformer/

Encoder

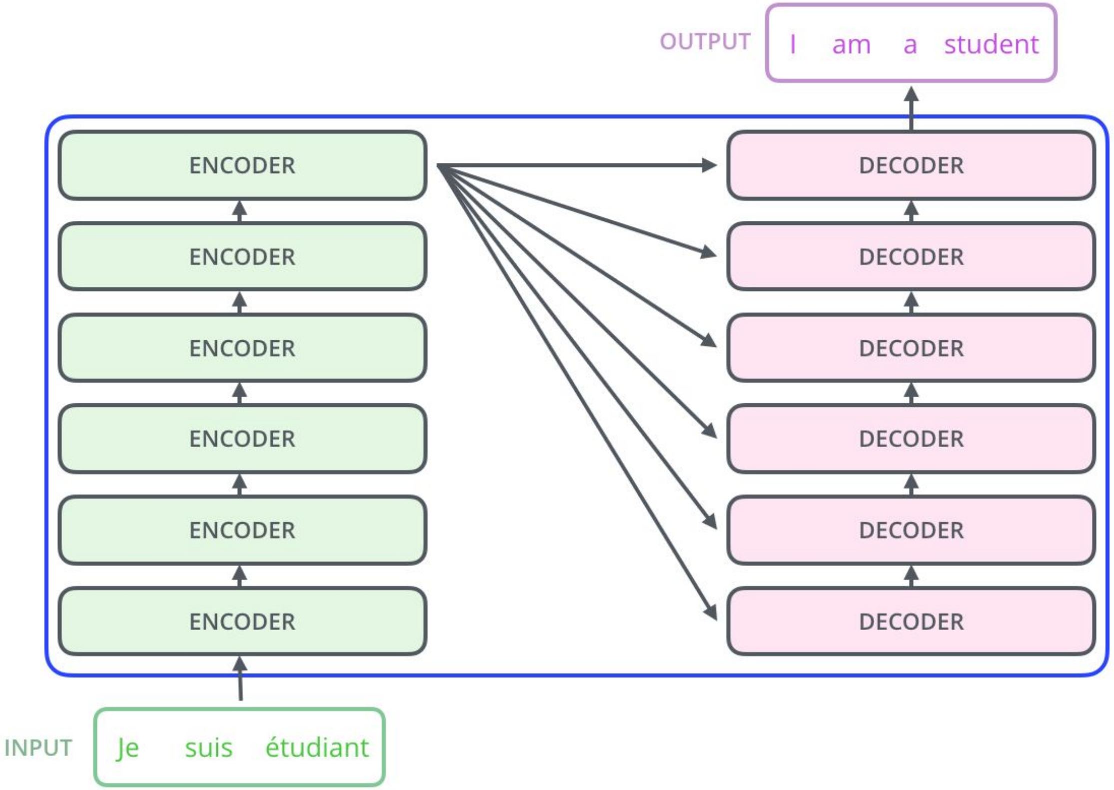


ENCODER #1
Add & Normalize
Feed Forward
Feed Forward
Add & Normalize
Self-Attention
POSITIONAL
ENCODING
x_1
x_2
Thinking
Machines

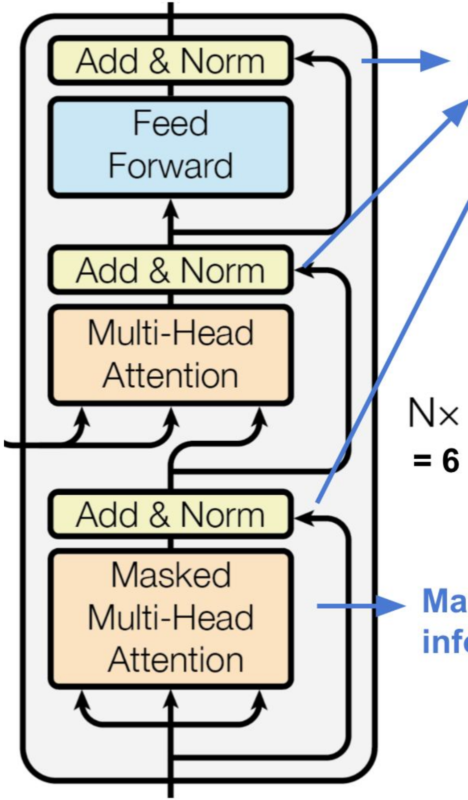


ENCODER #1
Add & Normalize
Feed Forward
Feed Forward
z_1
z_2
Add & Normalize
X
Z
LayerNorm(+)
z_1
z_2
Self-Attention
x_1
x_2
POSITIONAL
ENCODING
x_1
x_2
Thinking
Machines

Encoder\decoder



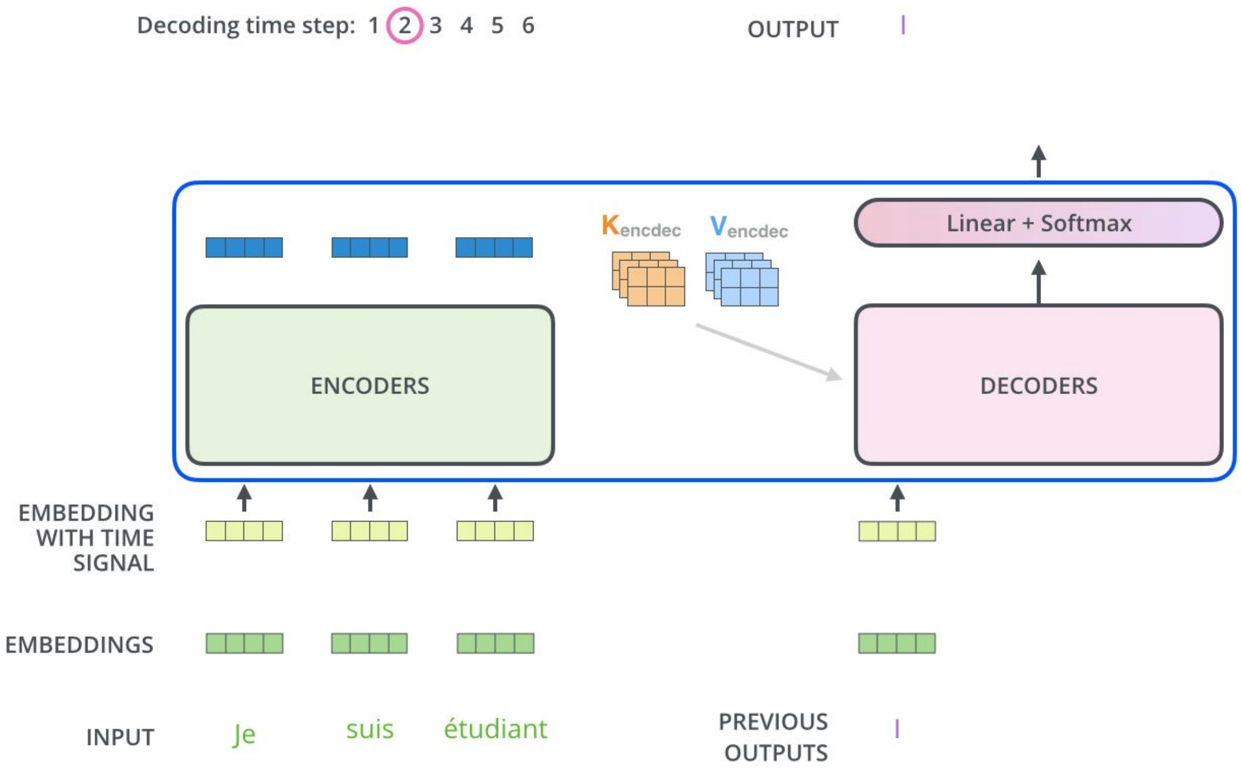
Decoder



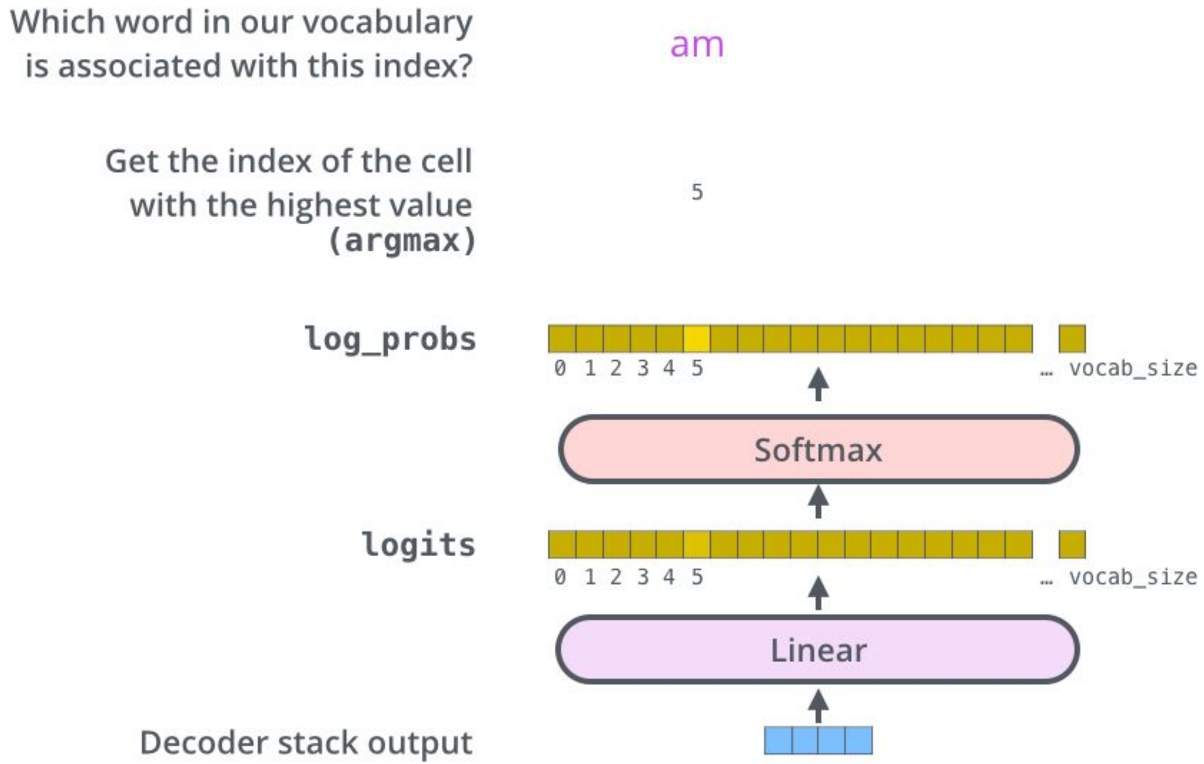
LayerNorm(x + SubLayer(x))

Masked: not to use the information in the future.

Decoder


Decoding time step: 1 2 3 4 5 6
OUTPUT
I
Kencdec
Vencdec
Linear + Softmax
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant
PREVIOUS OUTPUTS
I

Last layer


Which word in our vocabulary is associated with this index?
am
Get the index of the cell with the highest value (argmax)
5
log_probs
0 1 2 3 4 5 … vocab_size
Softmax
logits
0 1 2 3 4 5 … vocab_size
Linear
Decoder stack output

Attention is all you need

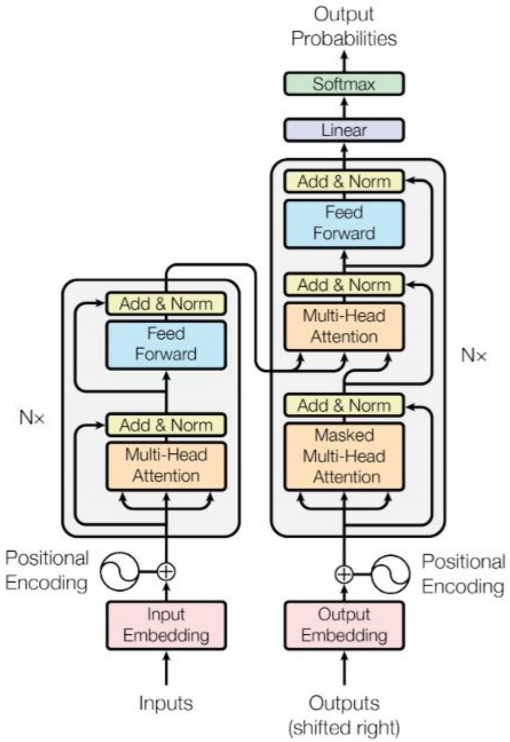
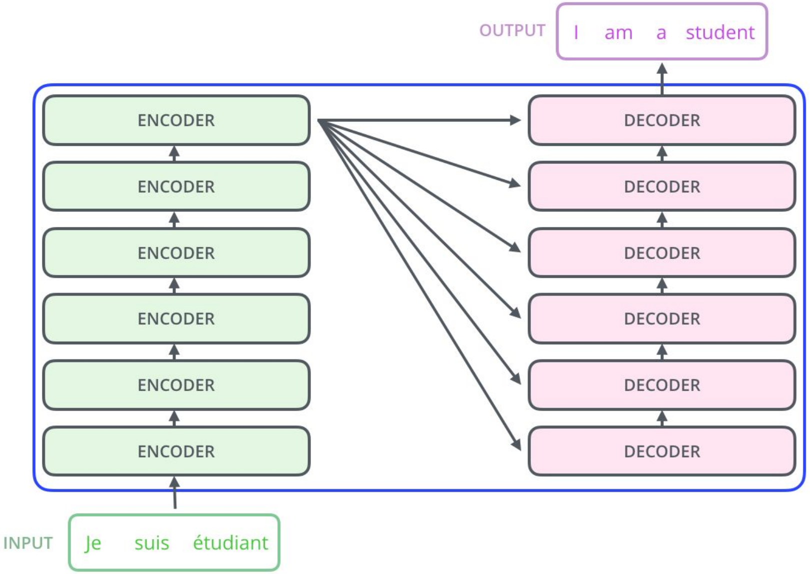
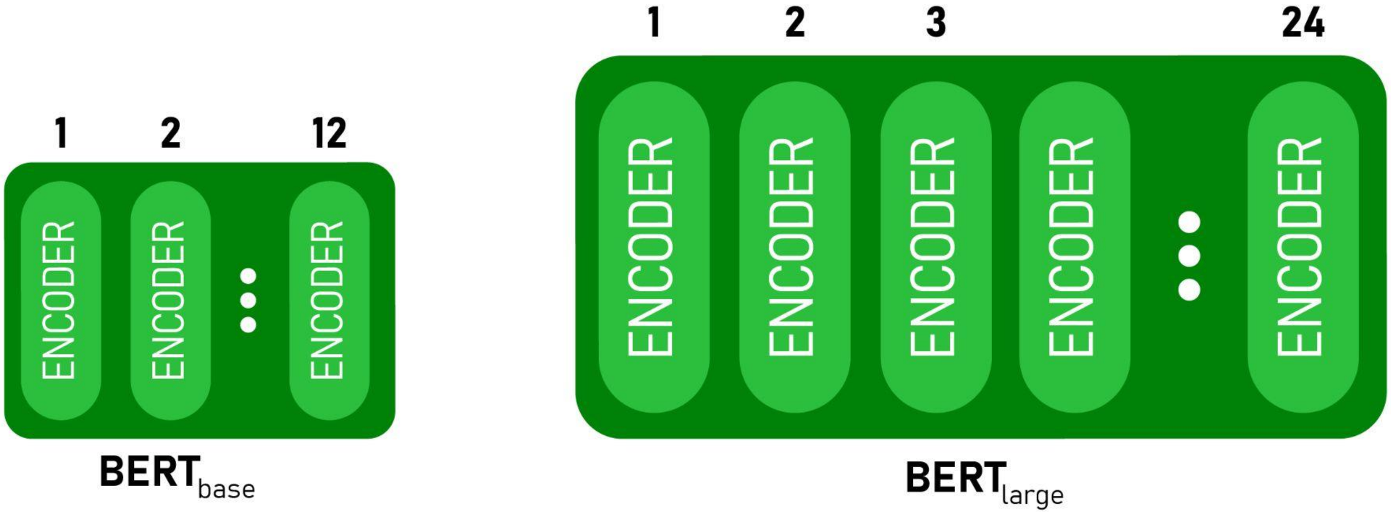


Figure 1: The Transformer - model architecture.

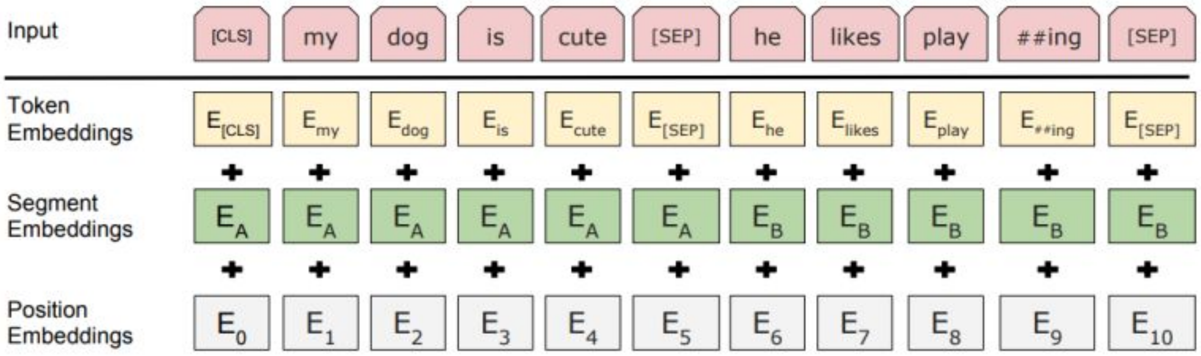


- no time loops
- can learn really long dependencies
- too many parameters

BERT - Bidirectional Encoding Representation from Transformers



Masked Language Modeling (MLM) and Next Sentence Prediction (NSP)



- 80% of the time the words were replaced with the masked token [MASK]
- 10% of the time the words were replaced with random words
- 10% of the time the words were left unchanged
- 50% of the time the second sentence comes after the first one.
- 50% of the time it is a random sentence from the full corpus.

Masked Language Modeling (MLM) and Next Sentence Prediction (NSP)

Input = [CLS] the man went to [MASK] store [SEP]

he bought a gallon [MASK] milk [SEP]

Label = IsNext

Input = [CLS] the man [MASK] to the store [SEP]

penguin [MASK] are flight ##less birds [SEP]

Label = NotNext

Recap

- Attention
- Applications
- Types of attention
- Transformer
  - Positional encoding
  - Self-attention
  - Multi-head attention
- BERT model (MLM)