Seq2seq, Attention, Self attention, Transformer, BERT

Tuan Nguyen - AI4E

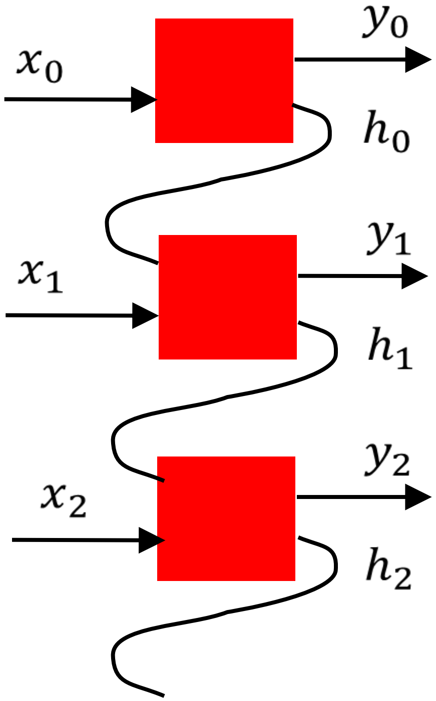
Outline

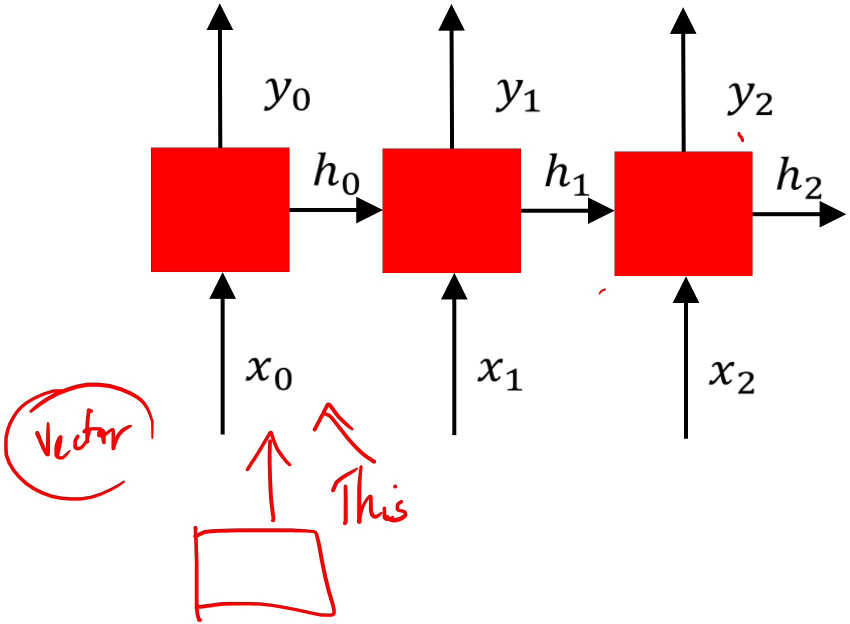
- RNN review
- Seq2seq
- Beam search
- Attention
- Self-attention
- Transformer
- BERT

Recurrent Neural Network

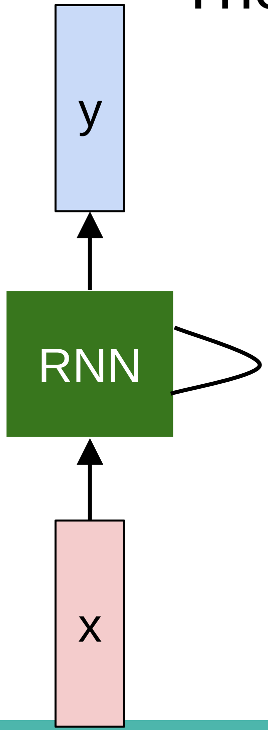
Sequence
time - series.

Usually drawn as:





RNN Formula



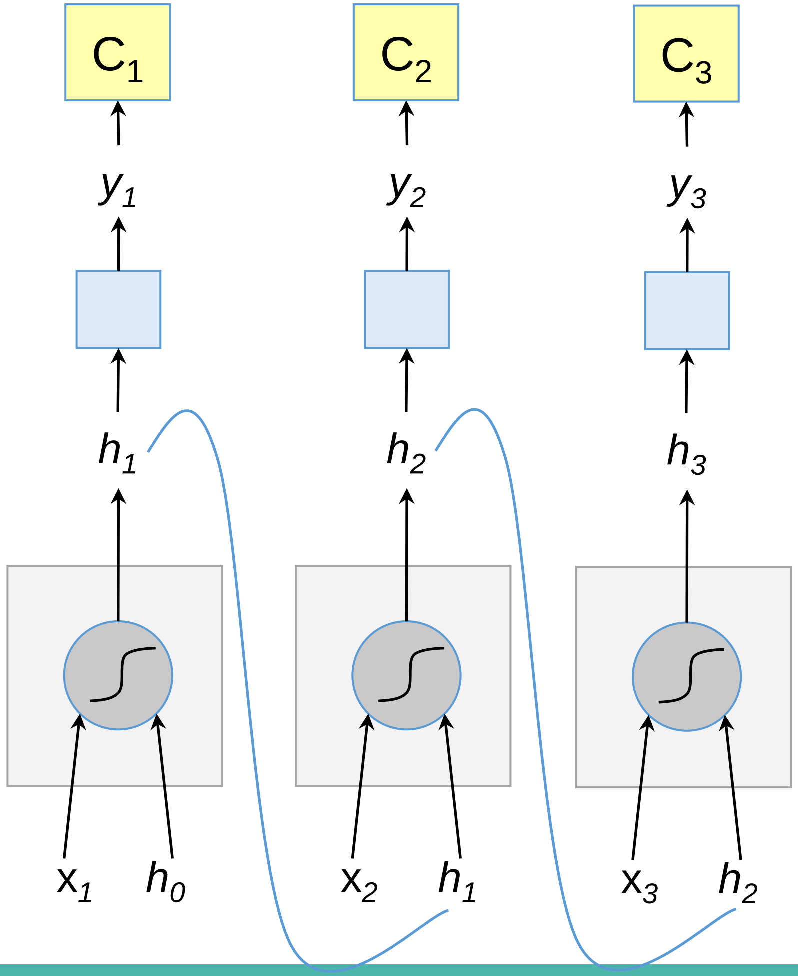
The state consists of a single *"hidden"* vector **h**:

$$h_t = f_W(h_{t-1}, x_t)$$

$$\downarrow$$

$$\left\{\begin{aligned} h_t &= \tanh(W_{hh}h_{t-1} + W_{xh}x_t) \\ y_t &= W_{hy}h_t \end{aligned}\right.$$

Forward

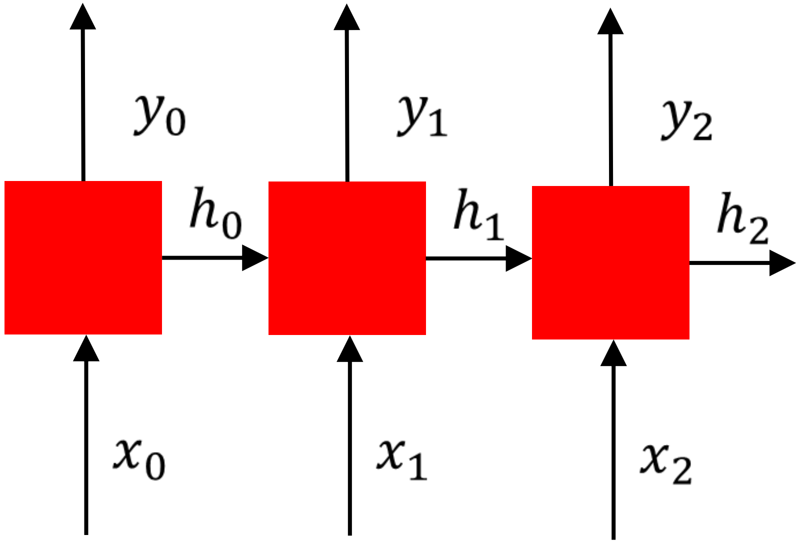


$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

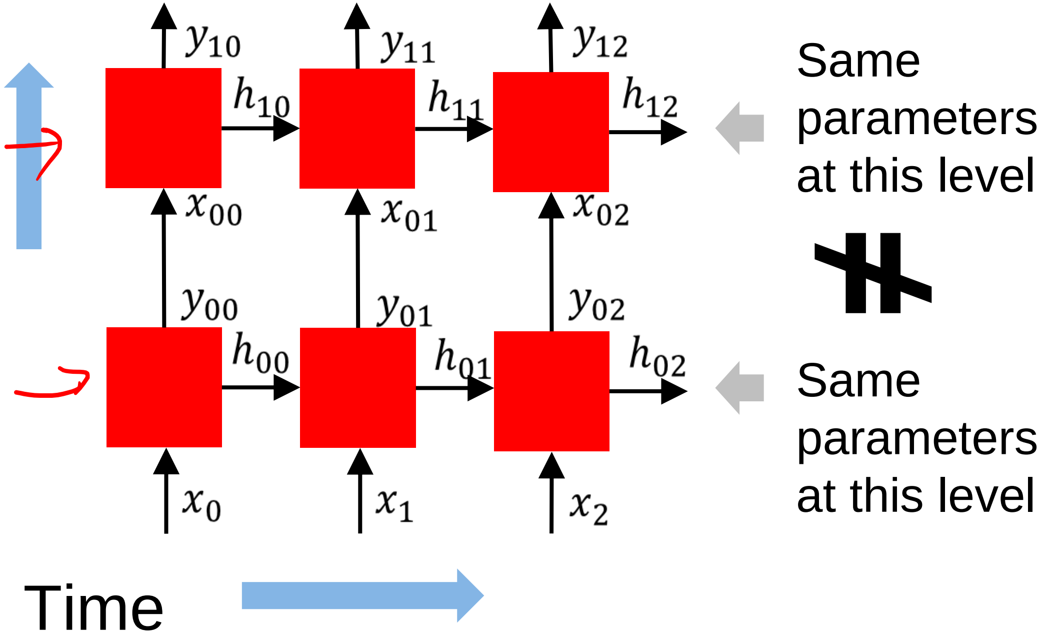
$$y_t = \mathrm{F}(h_t)$$

$$C_t = \mathrm{Loss}(y_t, \mathrm{GT}_t)$$

Deep RNN

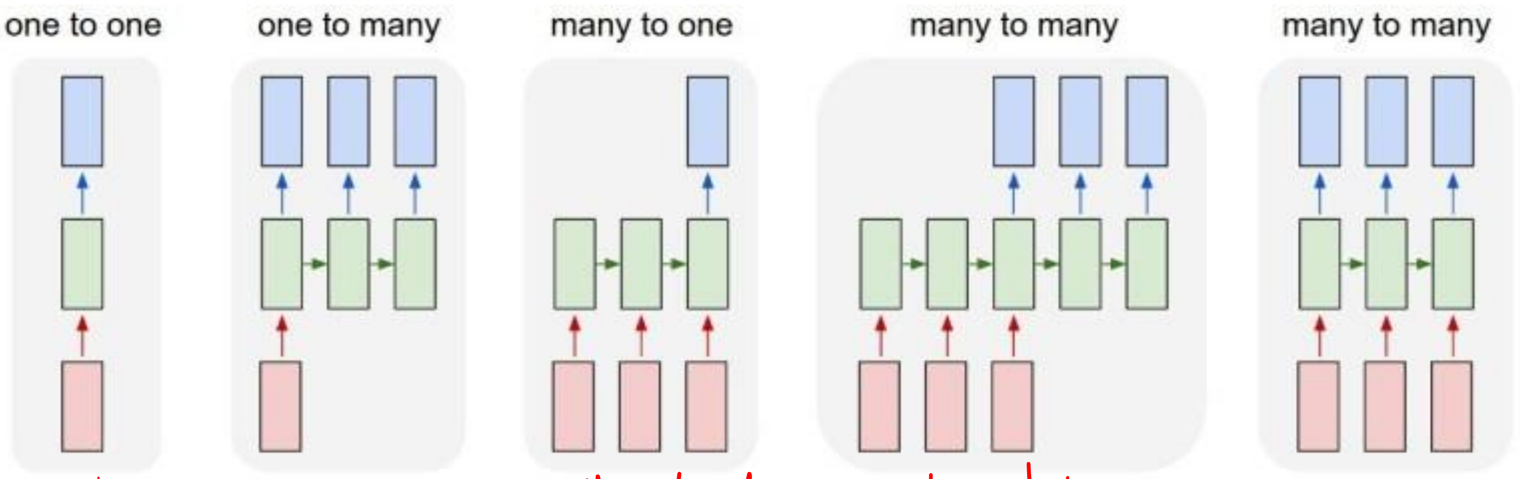


y_0
y_1
y_2
h_0
h_1
h_2
x_0
x_1
x_2



y_{10}
y_{11}
y_{12}
h_{10}
h_{11}
h_{12}
x_{00}
x_{01}
x_{02}
y_{00}
y_{01}
y_{02}
h_{00}
h_{01}
h_{02}
x_0
x_1
x_2
Time
Same parameters at this level
$\neq$
Same parameters at this level

Recurrent neural network problem



NN
CNN.

Image.
captioning

video classification

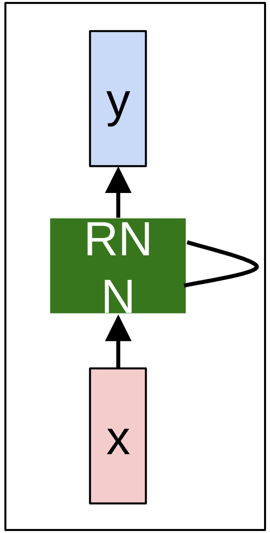
translation.

character generation.

Character-level language model example

Vocabulary:
[h,e,l,o]

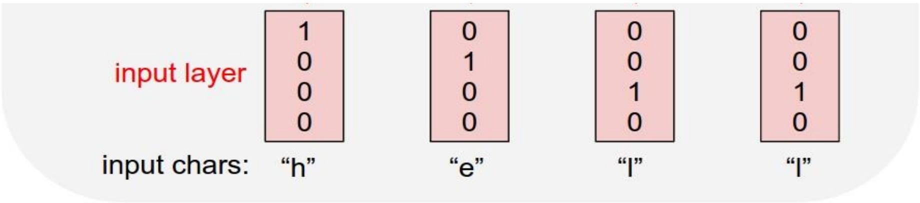
Example training sequence:
“hello”



Character-level language model example

Vocabulary:
[h,e,l,o]

Example trainin
sequence:
“hello”

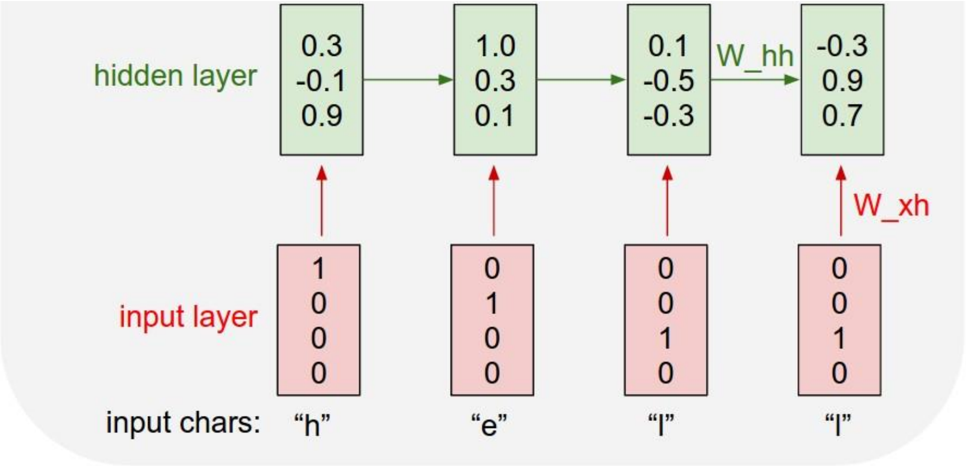


Character-level language model example

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”

$$\boxed{h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)}$$

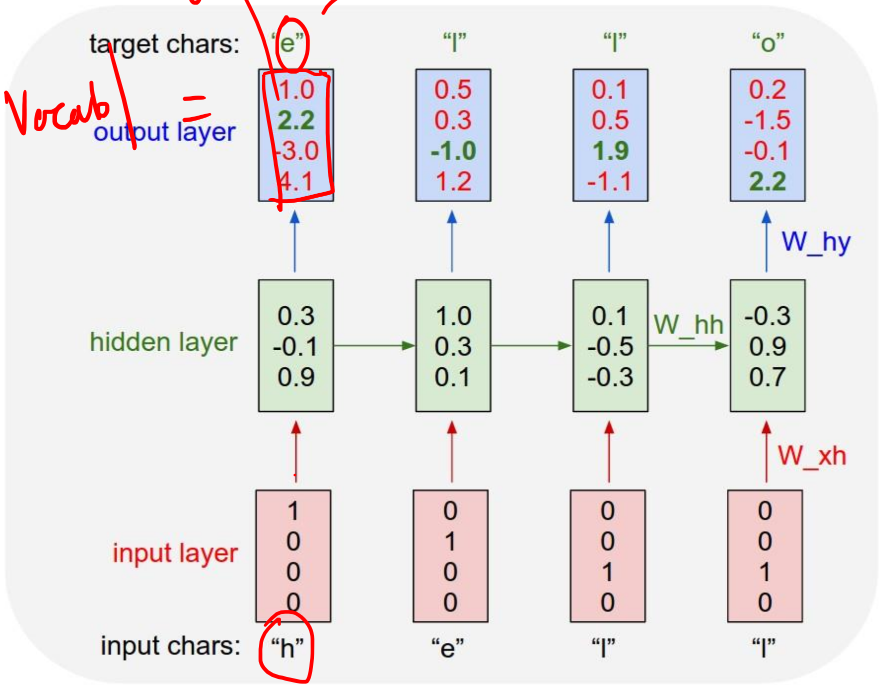


Character-level language model example

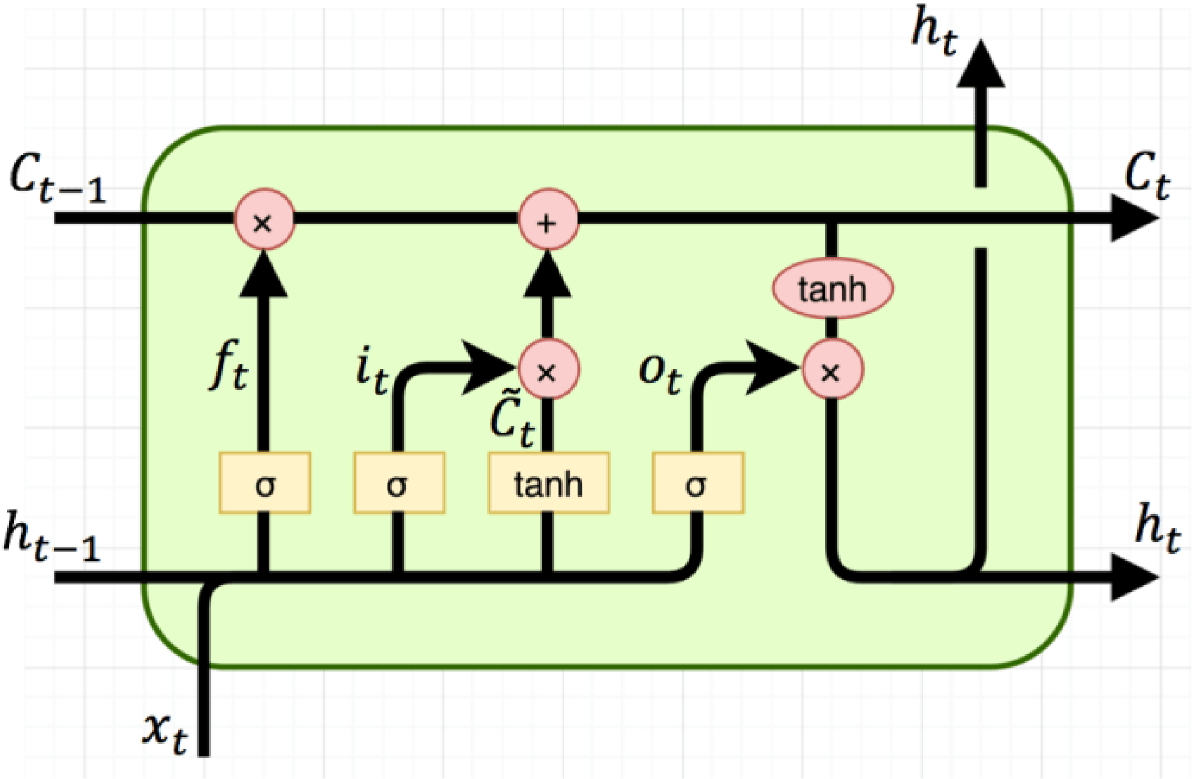
Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”

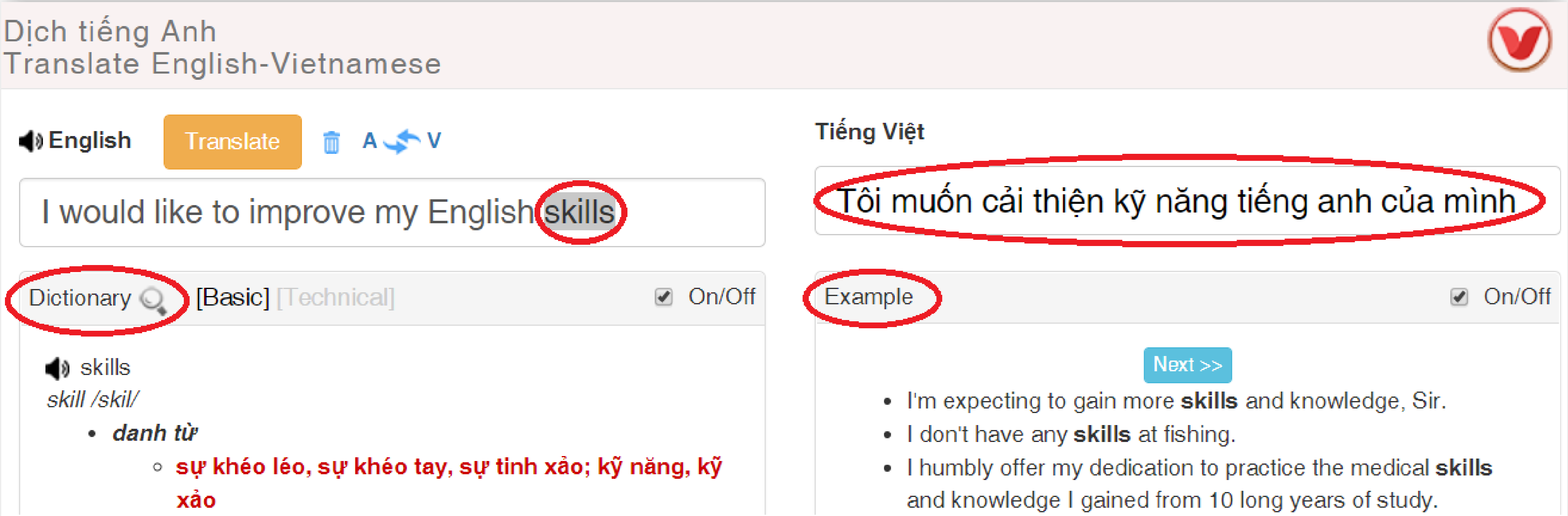
|Vocab| =



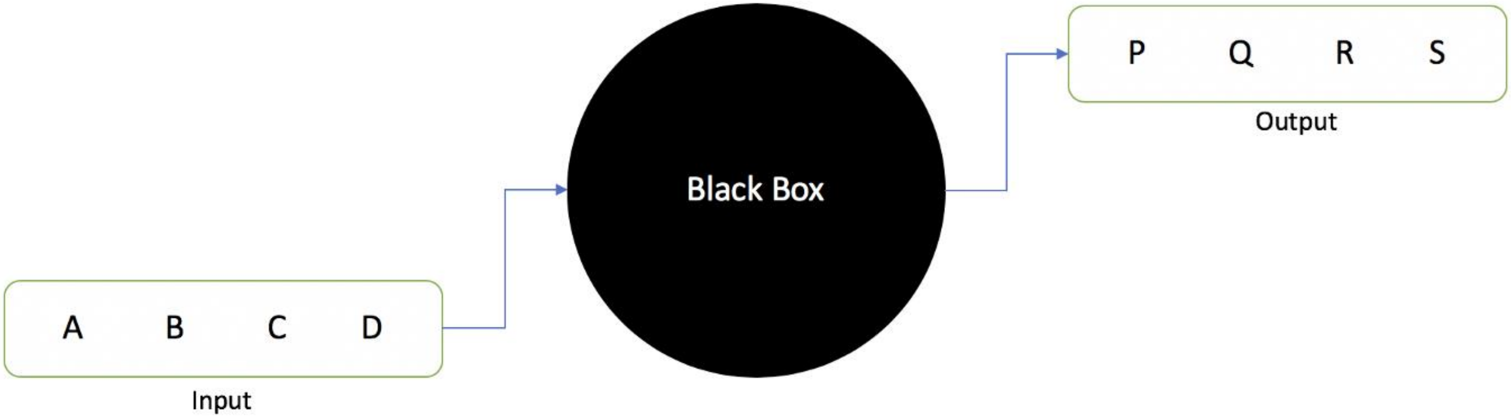
Long short term memory (LSTM) / GRU.



Translation



Model

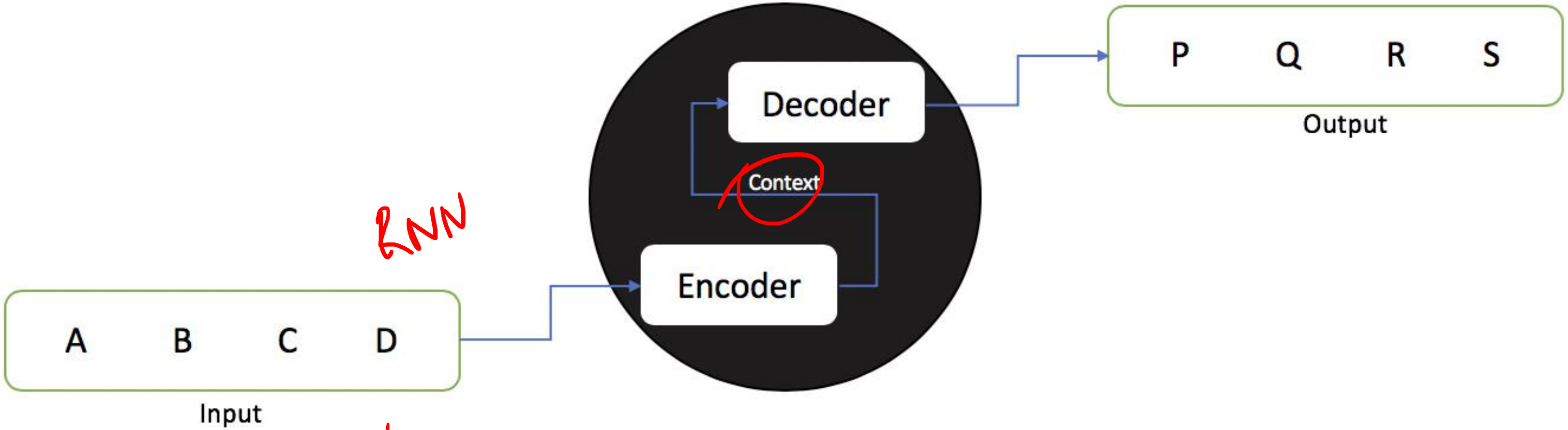


Seq2seq

autoencoder.

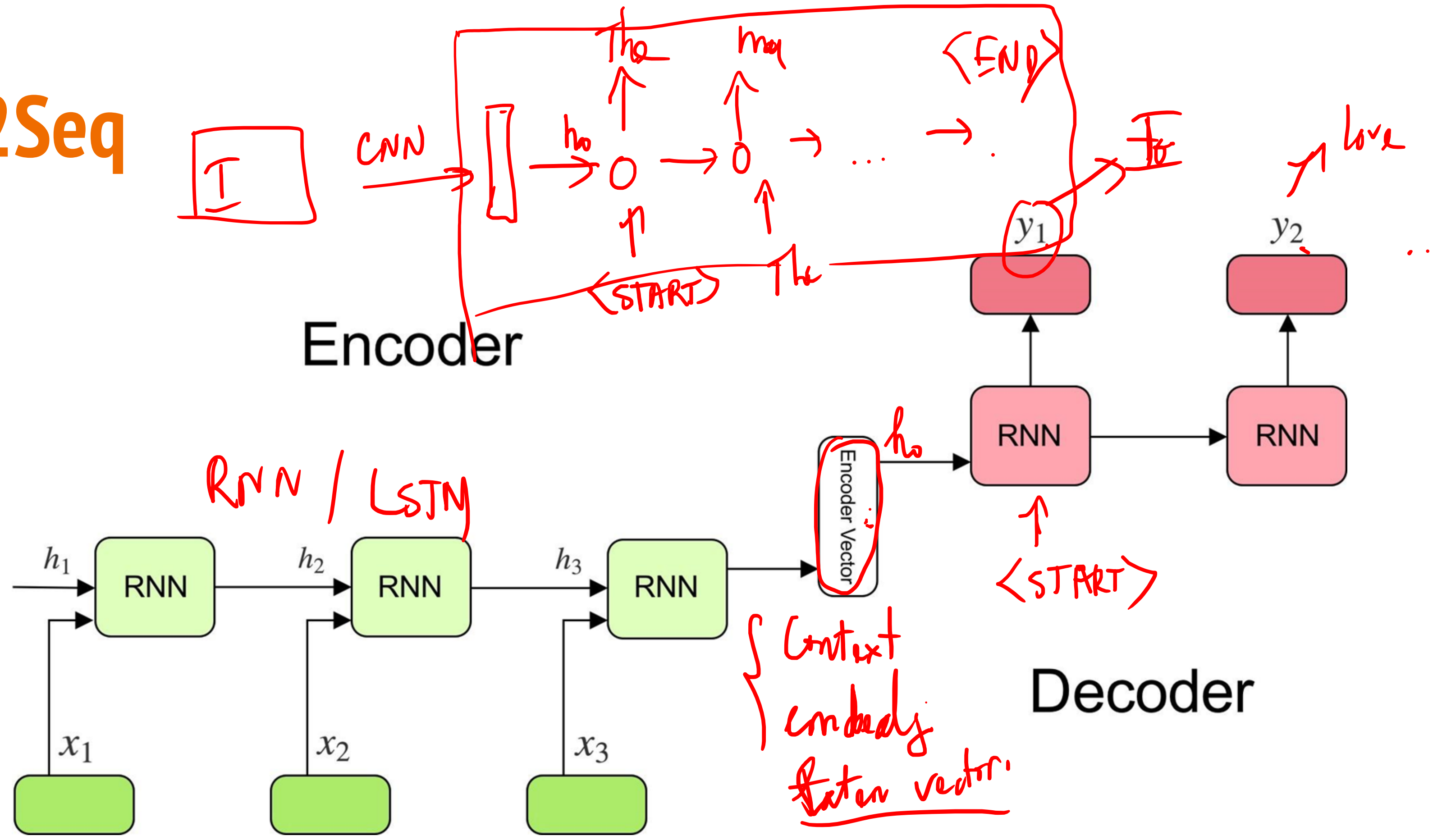
I love

RNN



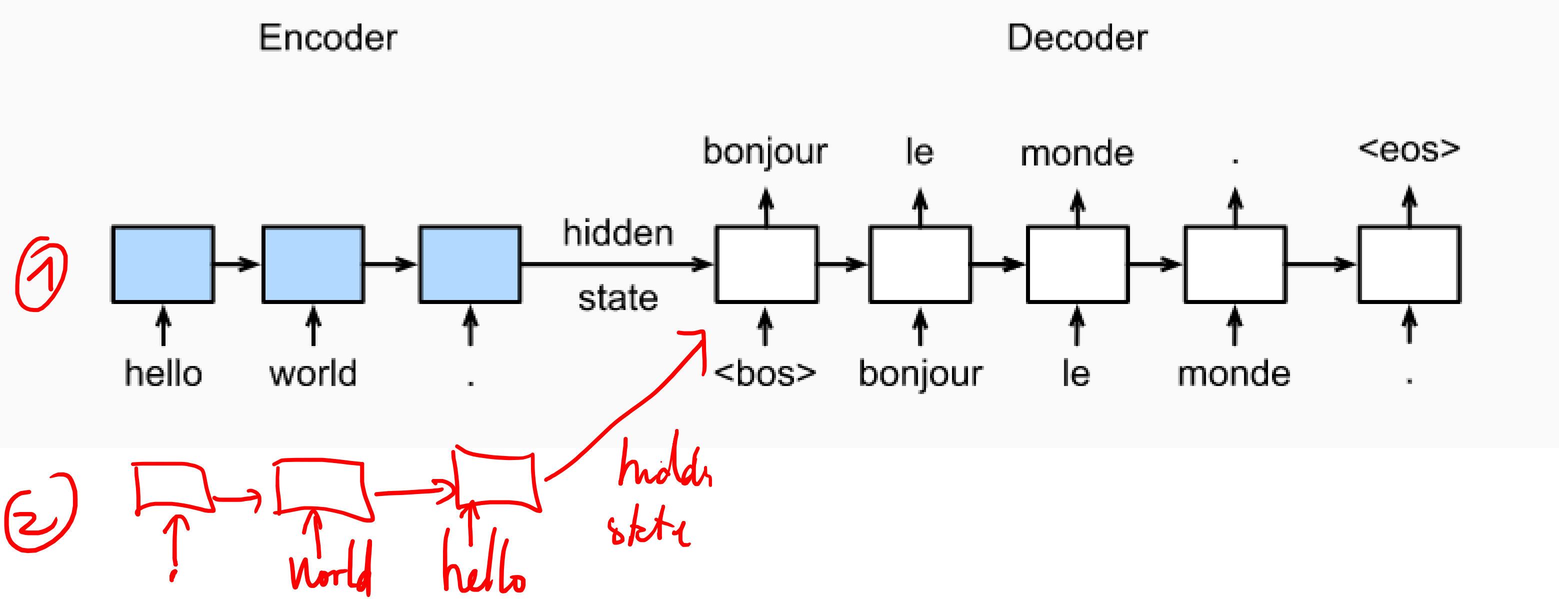
Tôi yêu Việt Nam

Seq2Seq

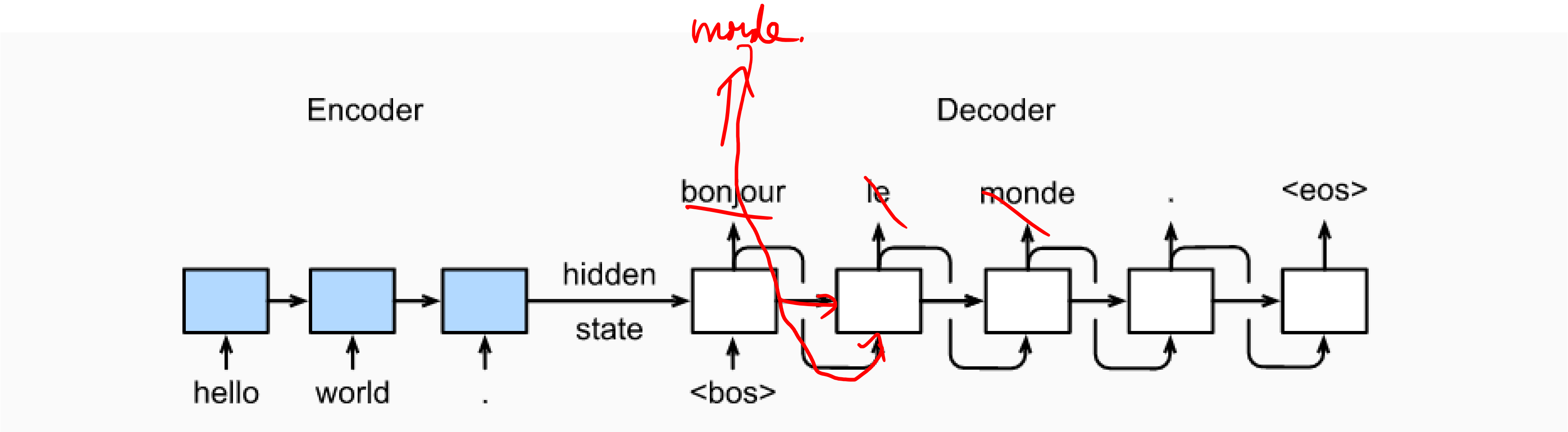


CNN
h_0
The
man
<END>
<START>
The
love
y_1
y_2
Encoder
RNN / LSTM
h_1
h_2
h_3
RNN
RNN
RNN
x_1
x_2
x_3
Encoder Vector
h_0
RNN
RNN
<START>
Context
embedding
latent vector
Decoder

Seq2seq


Encoder
Decoder
bonjour
le
monde
.
<eos>
hidden
state
hello
world
.
<bos>
bonjour
le
monde
.
1
2
world
hello
hidden
state

Seq2seq - prediction



Greedy search

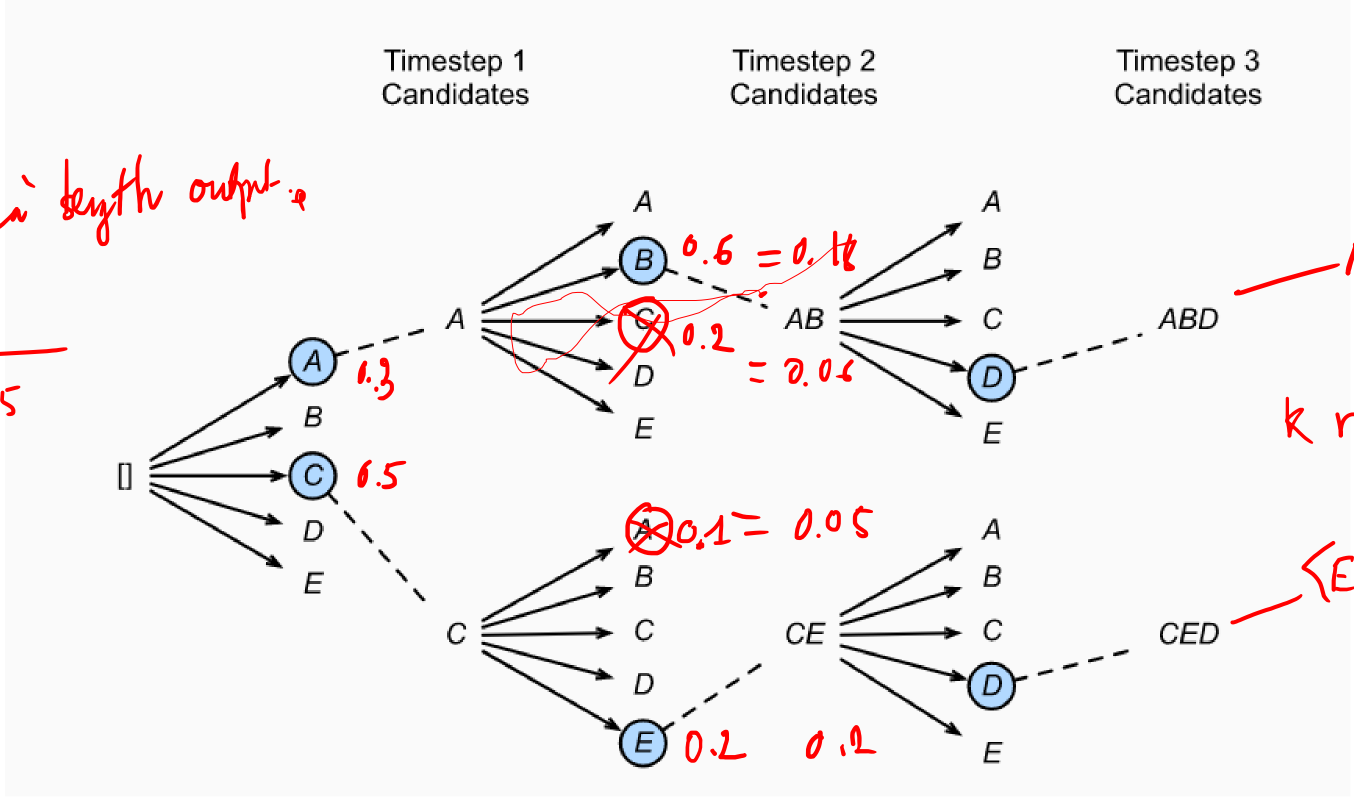
Timestep	1	2	3	4
A	0.5	0.1	0.2	0.0
B	0.2	0.4	0.2	0.2
C	0.2	0.3	0.4	0.2
<eos>	0.1	0.2	0.2	0.6

Beam search , $k = 2$, hyper-paramter giữ k nhánh tốt nhất

2^n, n là length output

$\sqrt[5]{x_1 \cdot x_2 \cdot x_3 \cdot x_4 \cdot x_5}$

$\sqrt[3]{y_1 \cdot y_2 \cdot y_3}$



ABDA — ABD<END>

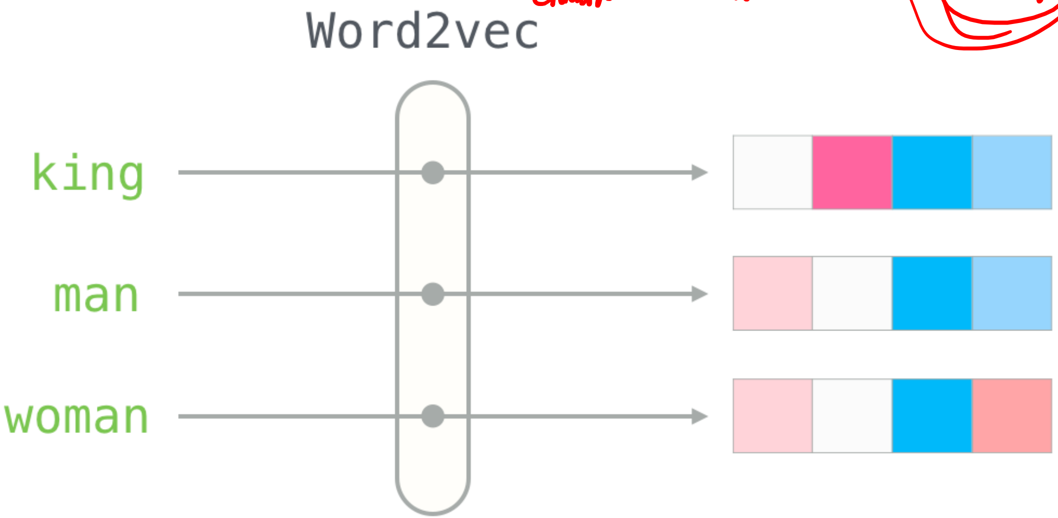
<END>

k nhánh.

Word embedding (glove, word2vec)

26 → 1,000 → 10,000

character word Sentence : 5 words

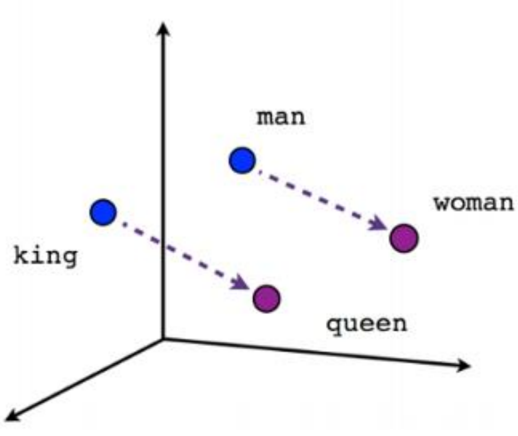


Word semantic

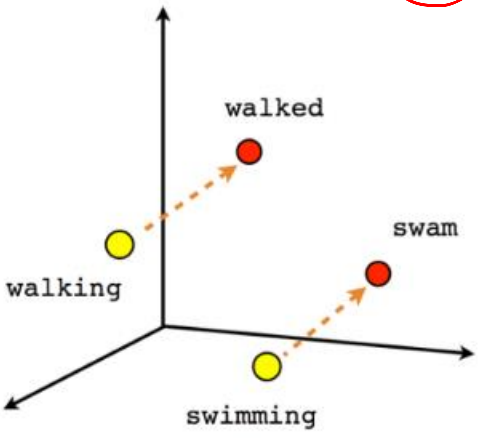
queen − king + man ≃ woman

Elmo

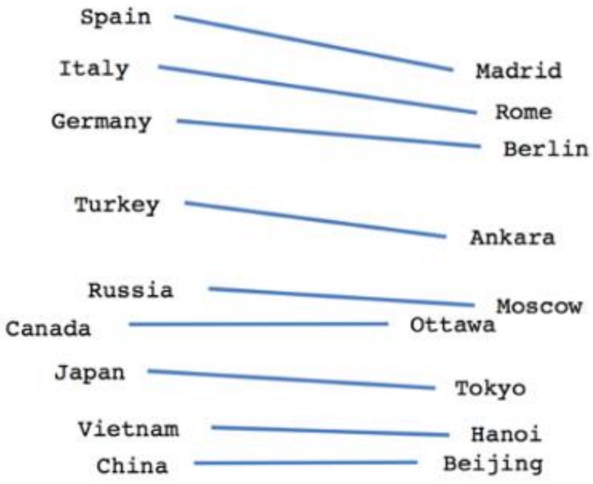
BERT



Male-Female



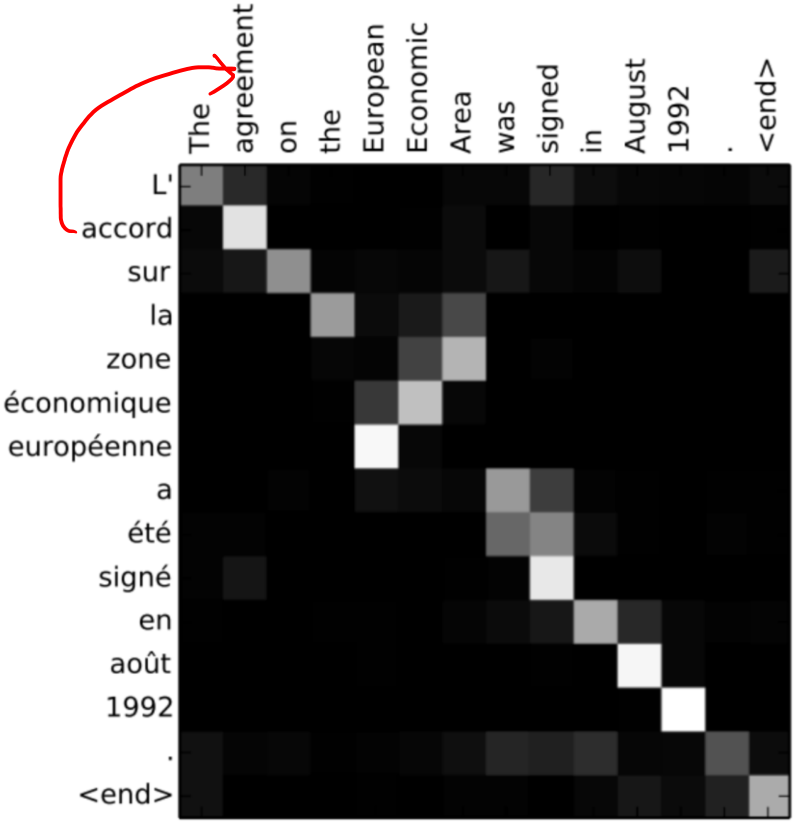
Verb tense



Country-Capital

Attention - motivation

The food is good → pos / neg



Attention

$V = \sum v_i \cdot b_i$

yêu love

q

$a_i = 2$

$k_1 = v_1$ $k_2 = v_2$ $k_3 = v_3$ $k_4 = v_4$

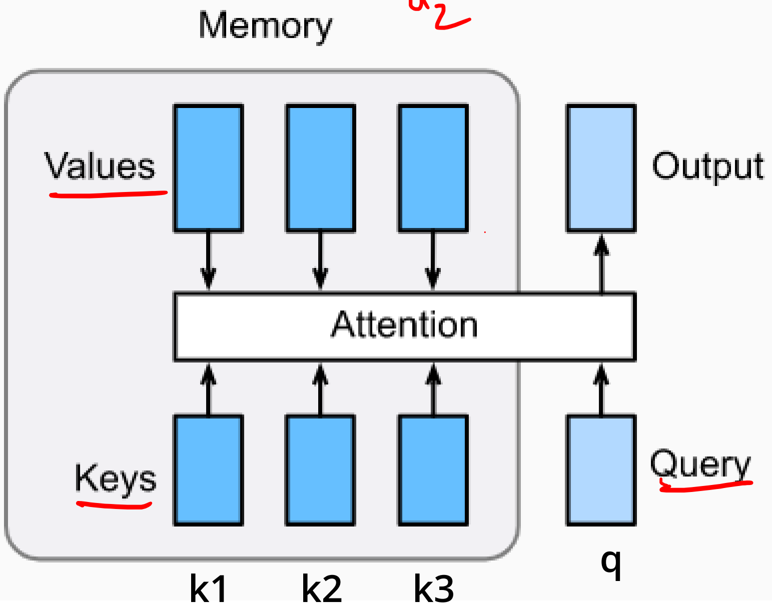
embedding từ Tôi yêu Việt Nam

a_2

$V = \sum v_i \cdot a_i$

$6 v_i$

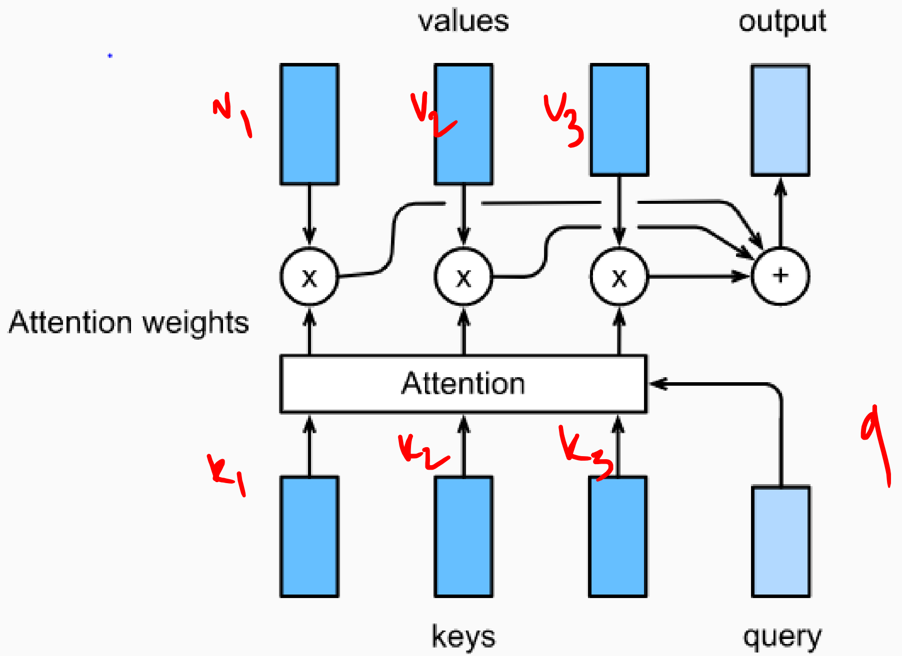
$\sum a_i > 1$



$a_i = \alpha(\mathbf{q}, \mathbf{k}_i)$: độ quan trọng từ i với từ cái đích

$$b_i = \frac{\exp(a_i)}{\sum_j \exp(a_j)}, \mathbf{b} = [b_1, \ldots, b_n]^T$$

Attention



values
output
v1
v2
v3
Attention weights
Attention
k1
k2
k3
q
keys
query

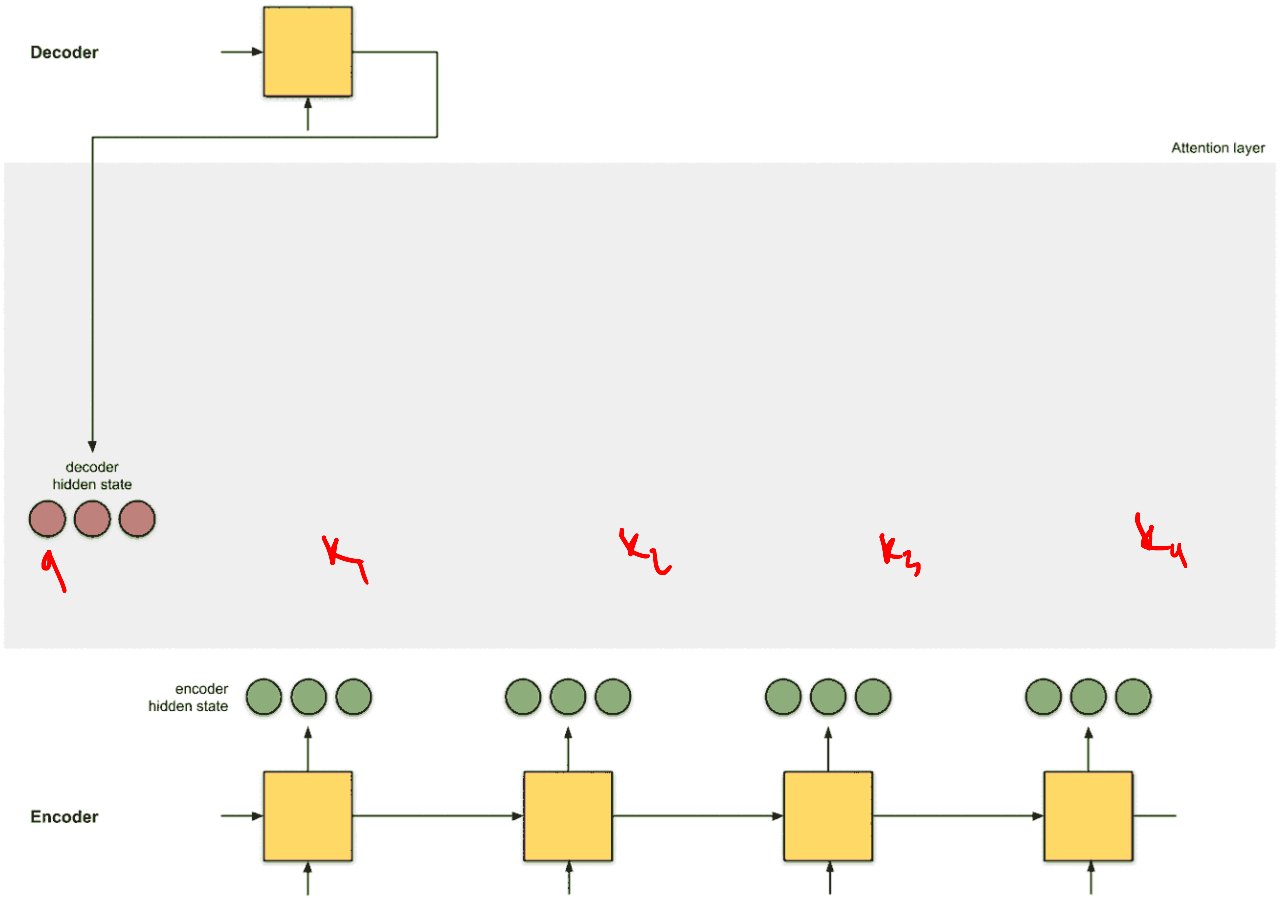
Attention function

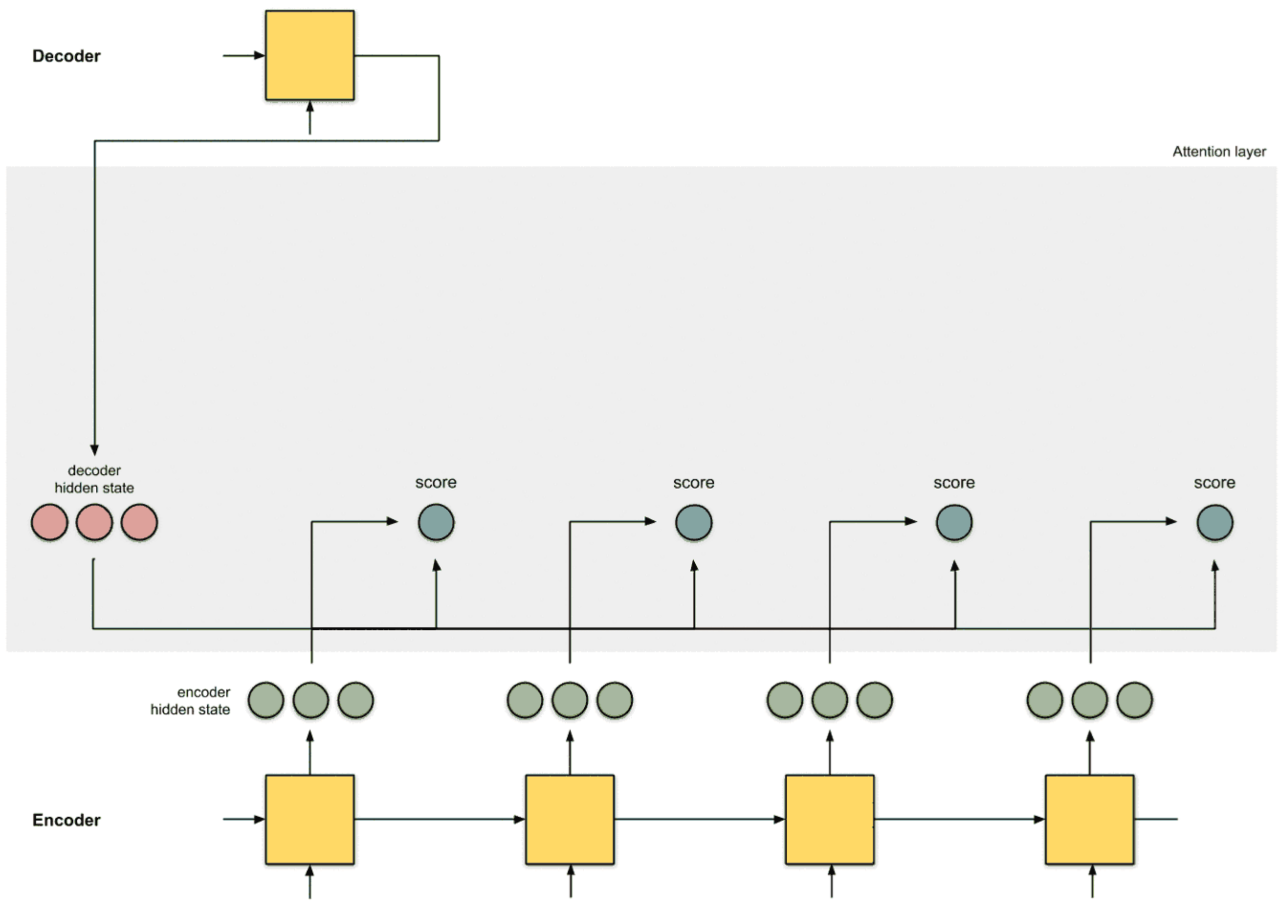
$a_i = \alpha(q, k_i)$

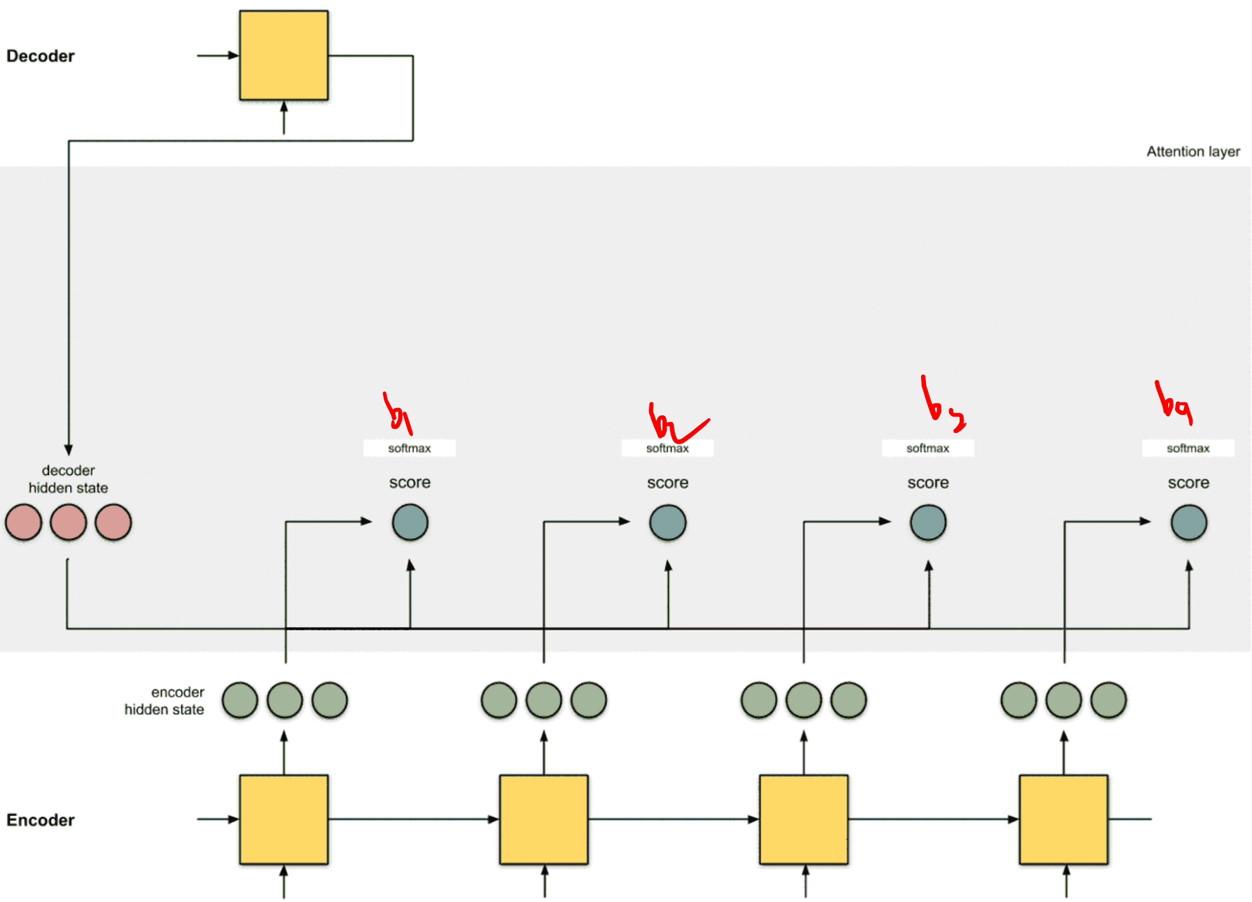
α: $q, k_i \rightarrow a_i$

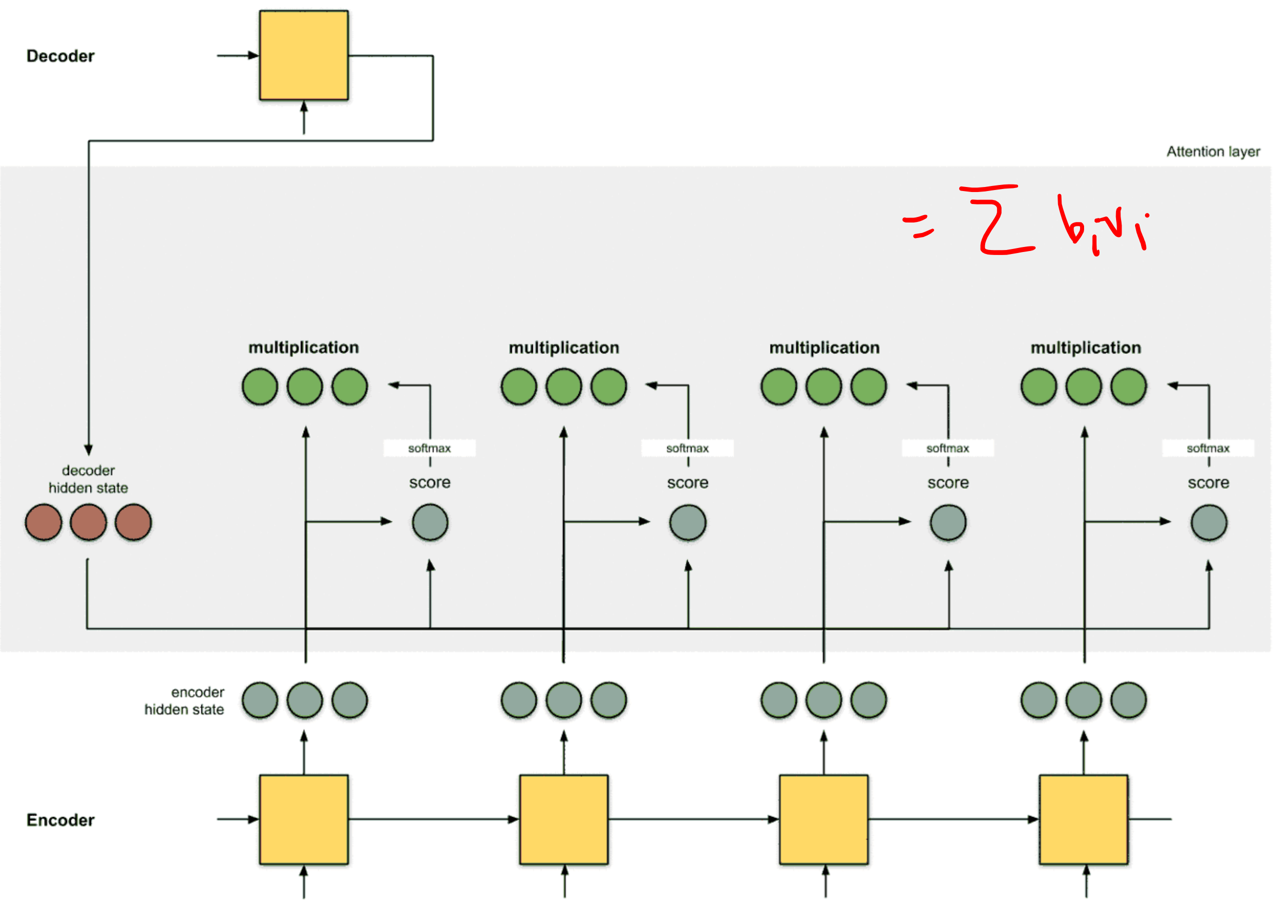
Name	Alignment score function
Content-base attention	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$
Additive(*)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$
Location-Base	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \boldsymbol{s}_t)$ Note: This simplifies the softmax alignment to only depend on the target position.
General	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \mathbf{W}_a \boldsymbol{h}_i$ where $\mathbf{W}_a$ is a trainable weight matrix in the attention layer.
Dot-Product	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \boldsymbol{h}_i$
Scaled Dot-Product(^)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \frac{\boldsymbol{s}_t^\top \boldsymbol{h}_i}{\sqrt{n}}$ Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state.

Encoder

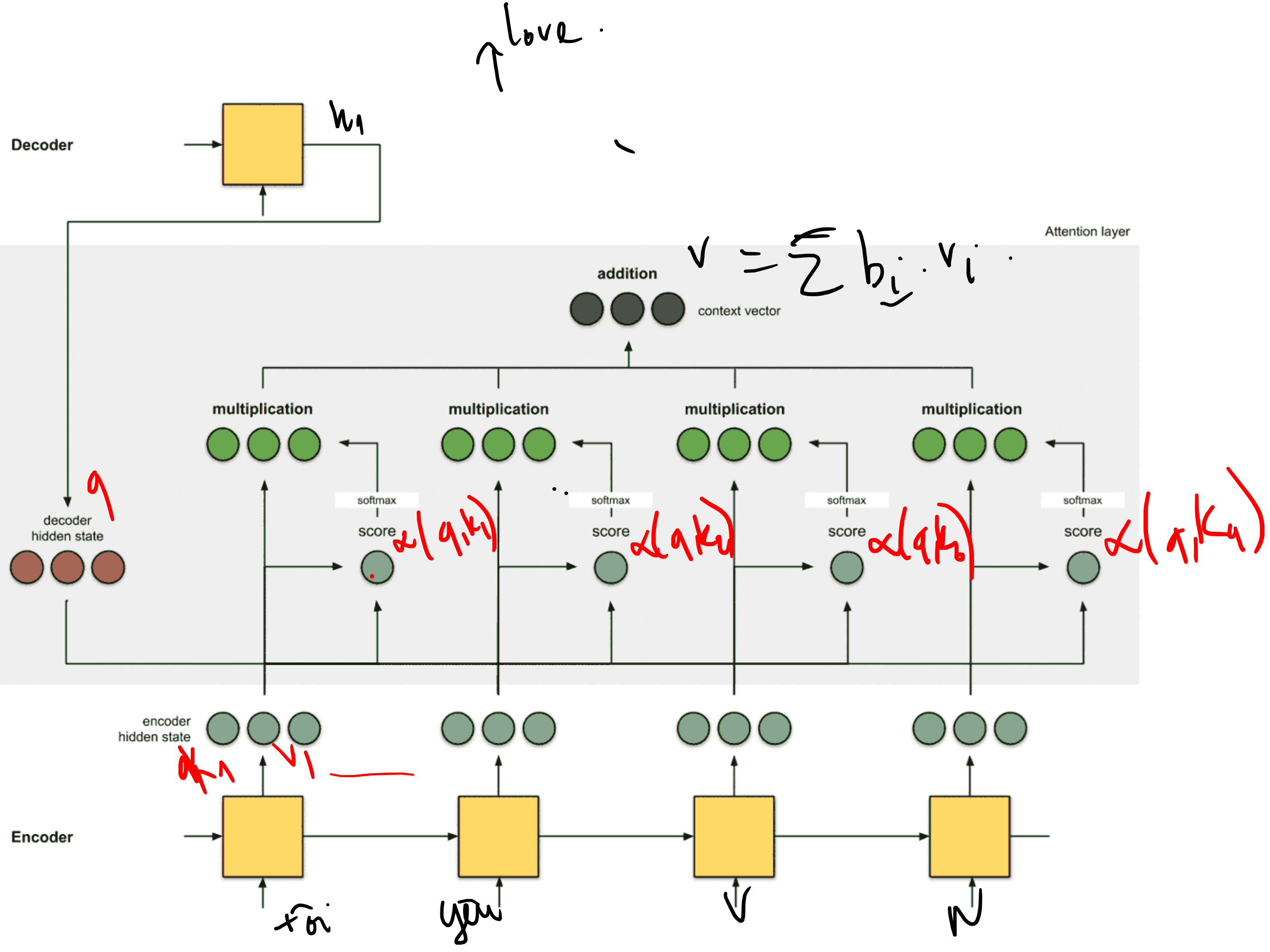

Decoder
Attention layer
decoder
hidden state
q
k1
k2
k3
k4
encoder
hidden state
Encoder


Decoder
Attention layer
decoder
hidden state
score
score
score
score
encoder
hidden state
Encoder


Decoder
Attention layer
b1
b2
b3
b4
softmax
softmax
softmax
softmax
score
score
score
score
decoder
hidden state
encoder
hidden state
Encoder

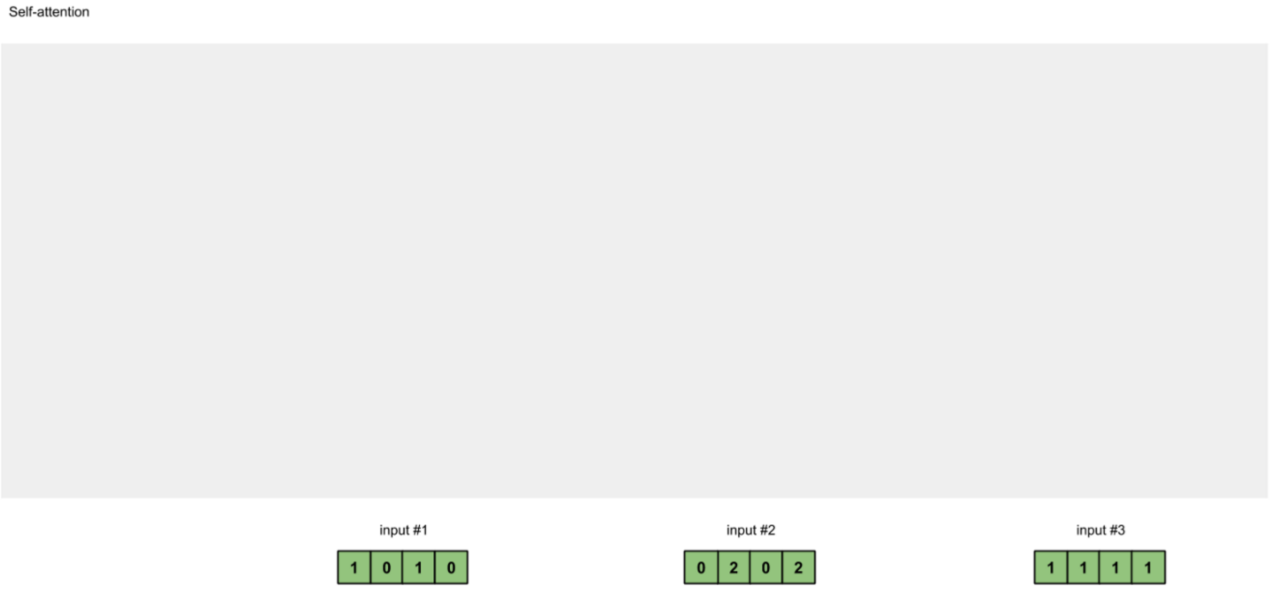


Decoder
Attention layer
$= \sum b_i v_i$
multiplication
multiplication
multiplication
multiplication
softmax
softmax
softmax
softmax
score
score
score
score
decoder
hidden state
encoder
hidden state
Encoder



love
Decoder
h1
Attention layer
v = Σ b_i · v_i
addition
context vector
multiplication
multiplication
multiplication
multiplication
softmax
softmax
softmax
softmax
score
score
score
score
q
decoder
hidden state
α(q,k1)
encoder
hidden state
k1
v1
Encoder
you

Self attention : học high-level feature của câu.



Self-attention

256 . 256 .

K, Q, V 128

K, 256

128

256 .

input #1

1	0	1	0

I

input #2

0	2	0	2

love

input #3

1	1	1	1

you

Self-attention

input #1

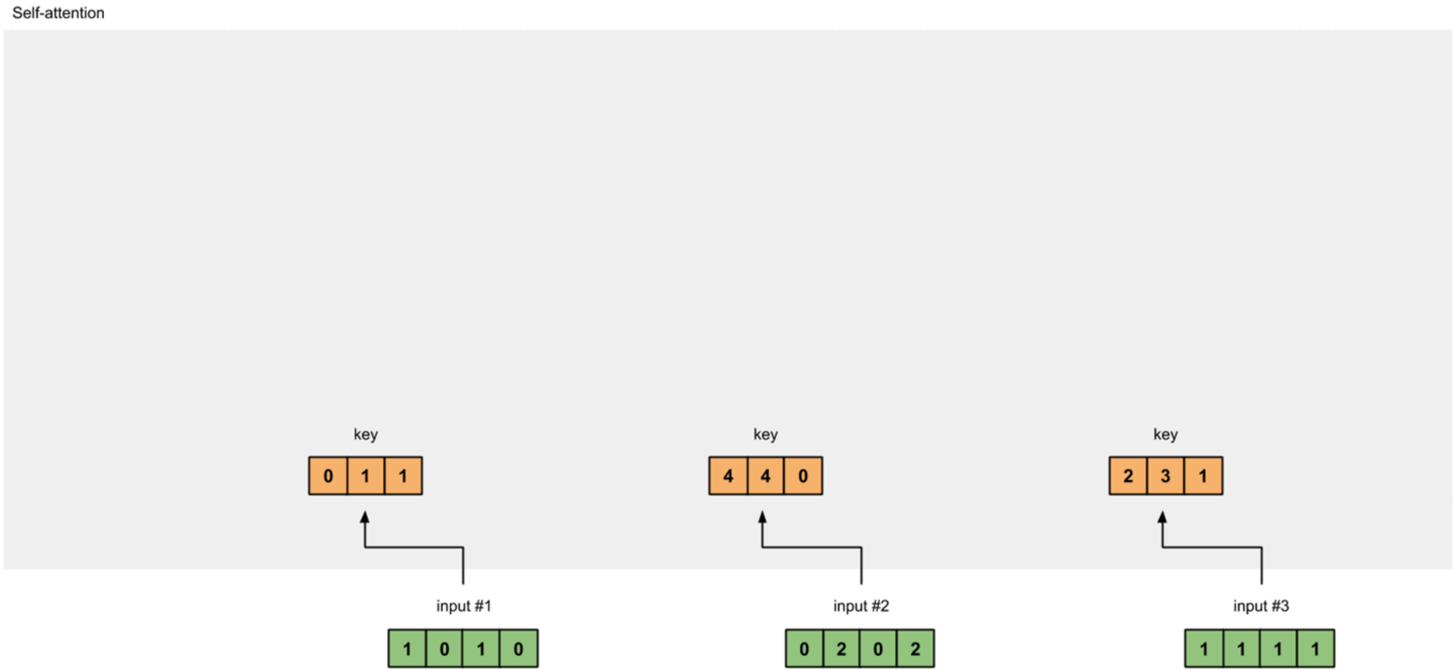
1	0	1	0

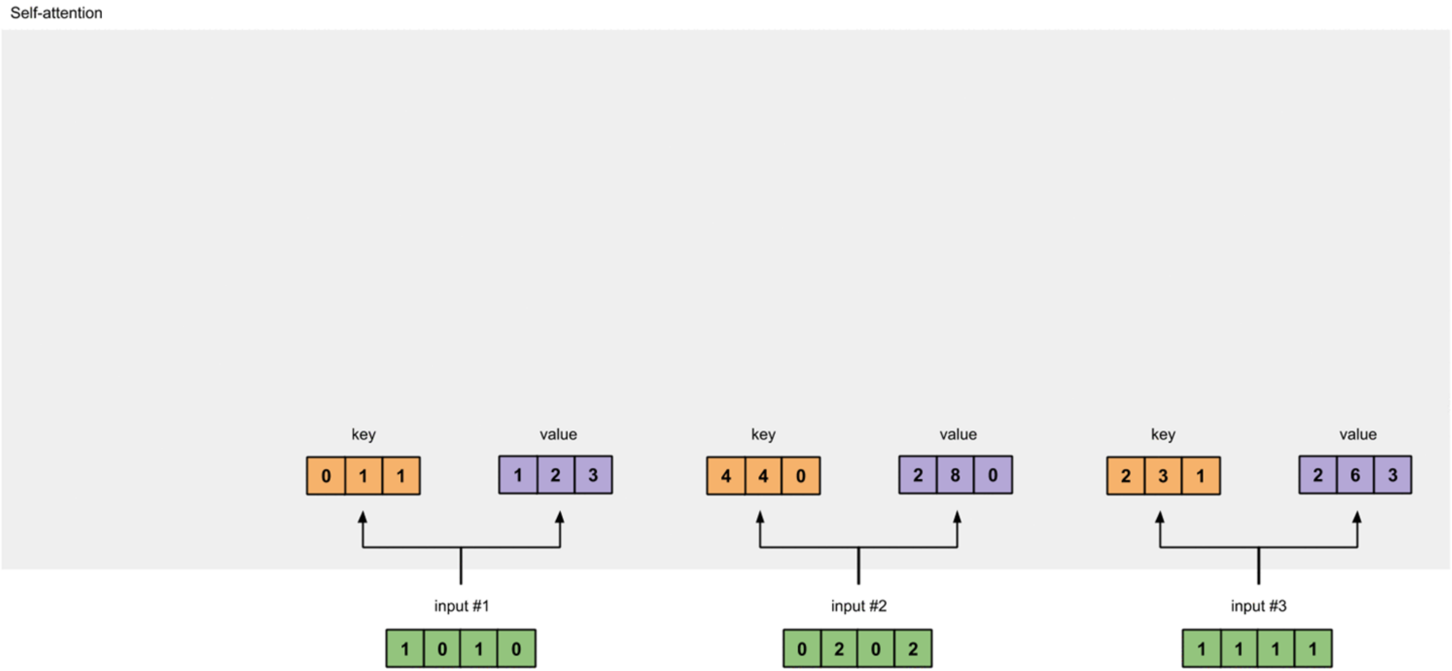
input #2

0	2	0	2

input #3

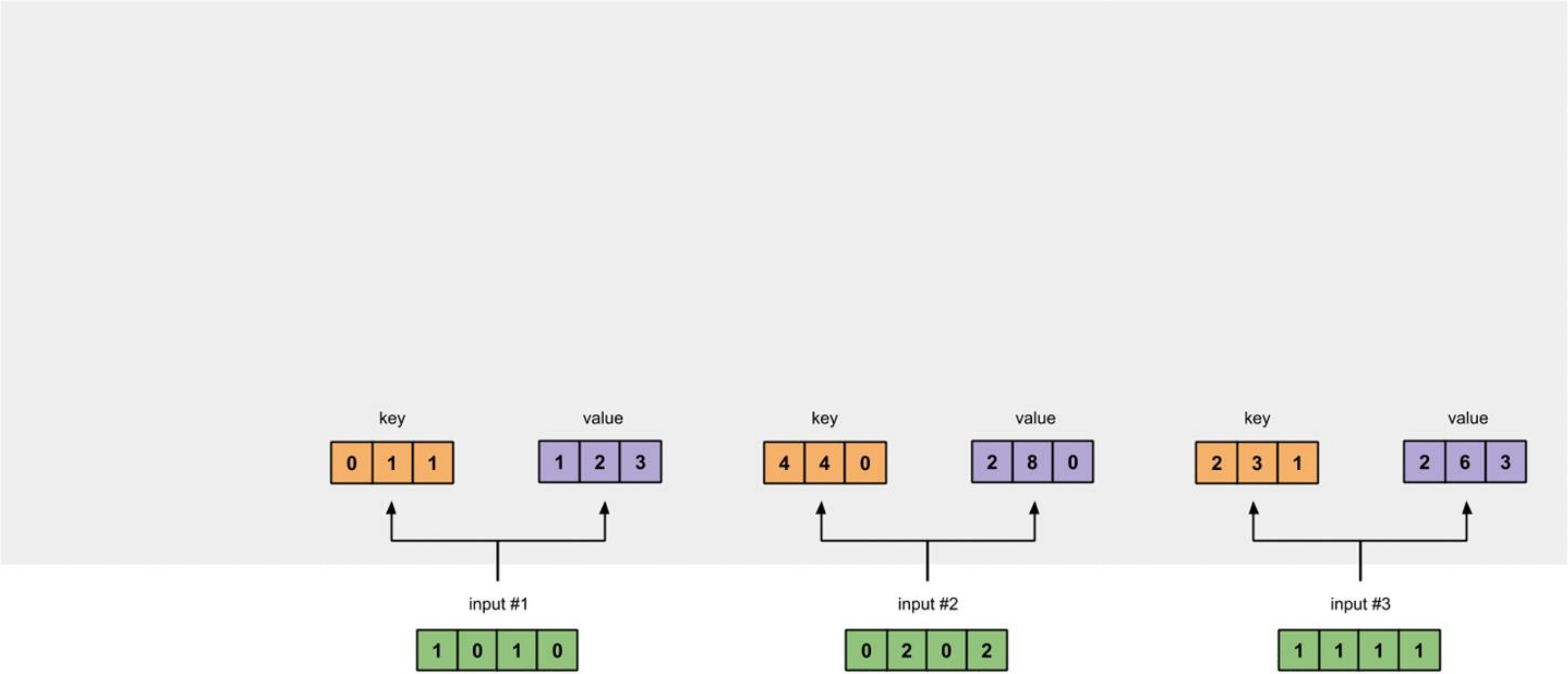
1	1	1	1


Self-attention
key
0 1 1
key
4 4 0
key
2 3 1
input #1
1 0 1 0
input #2
0 2 0 2
input #3
1 1 1 1

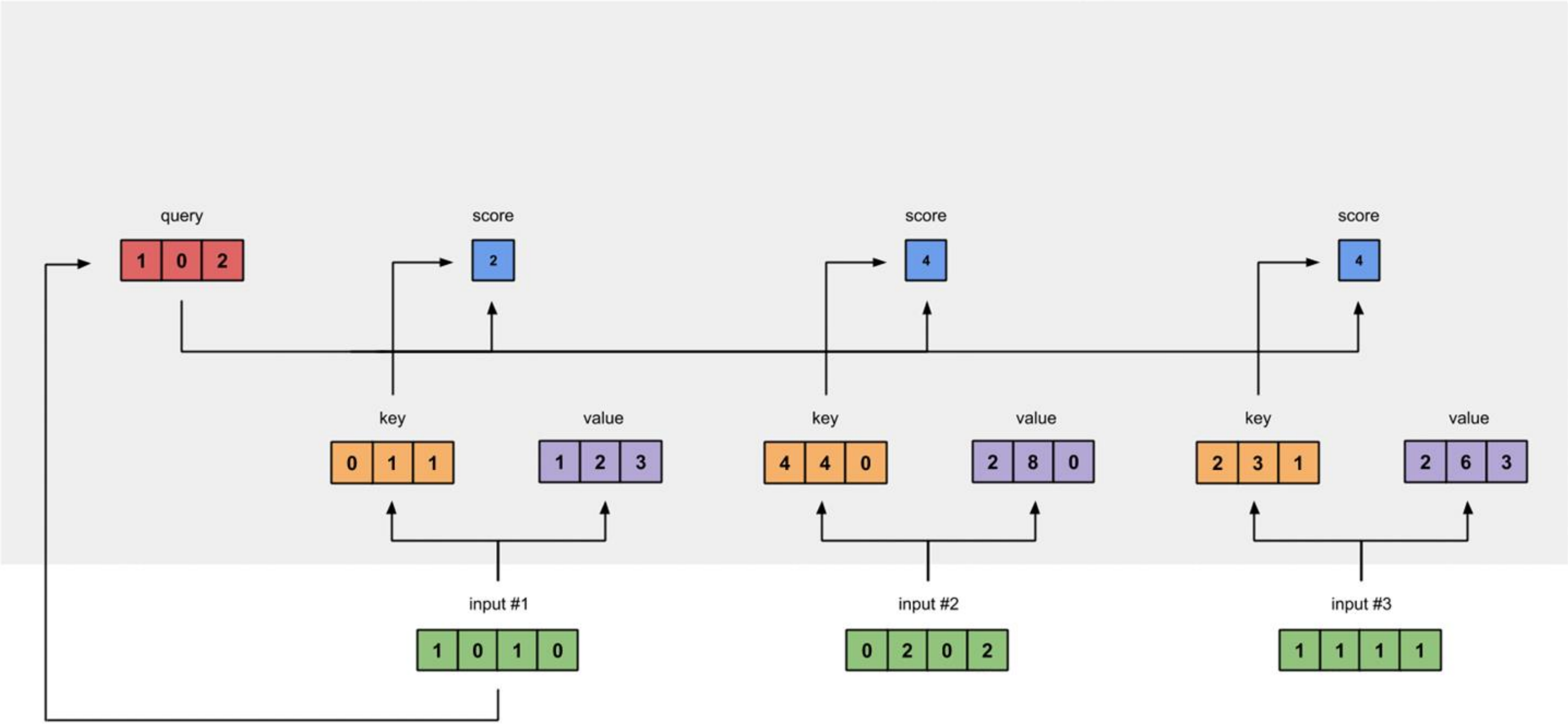


Self-attention
key
0 1 1
value
1 2 3
key
4 4 0
value
2 8 0
key
2 3 1
value
2 6 3
input #1
1 0 1 0
input #2
0 2 0 2
input #3
1 1 1 1

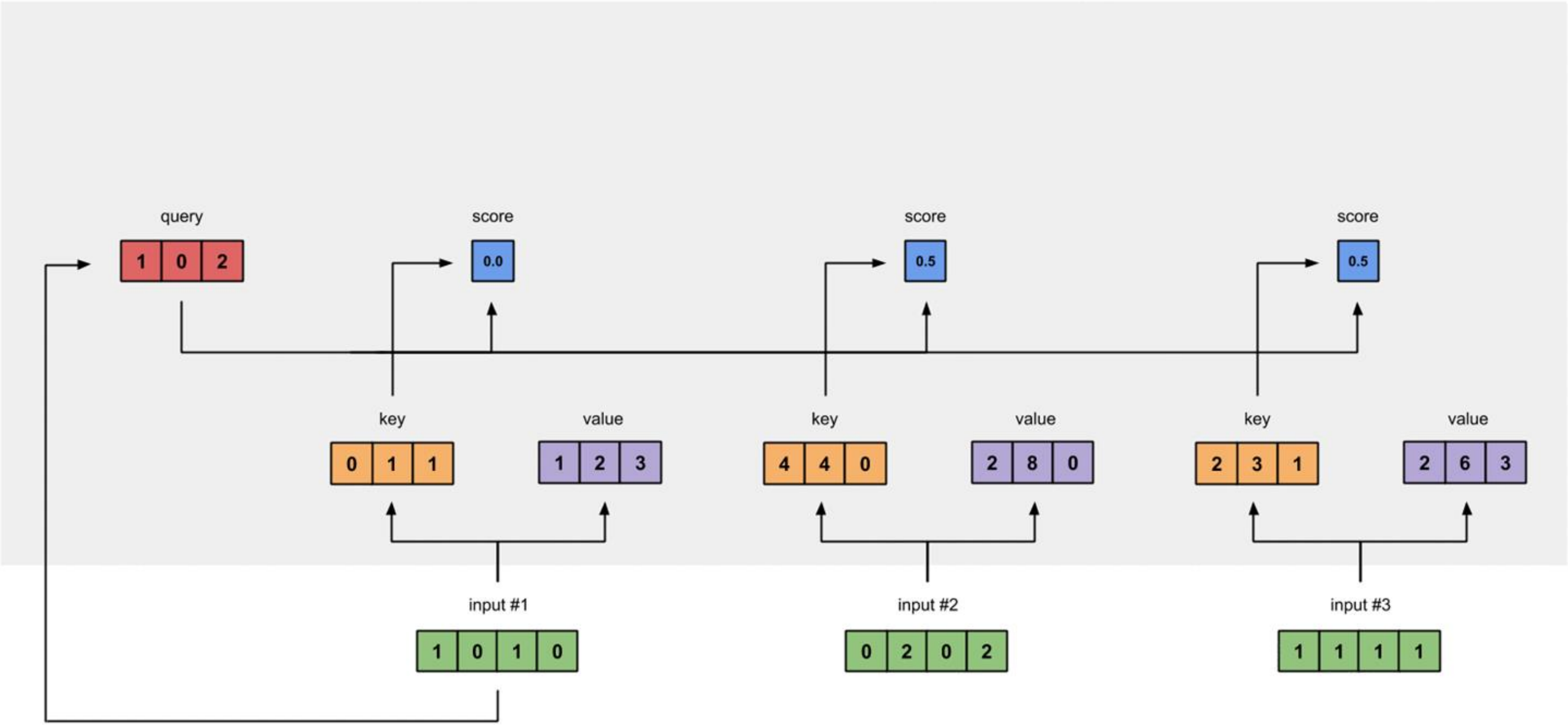
Self-attention



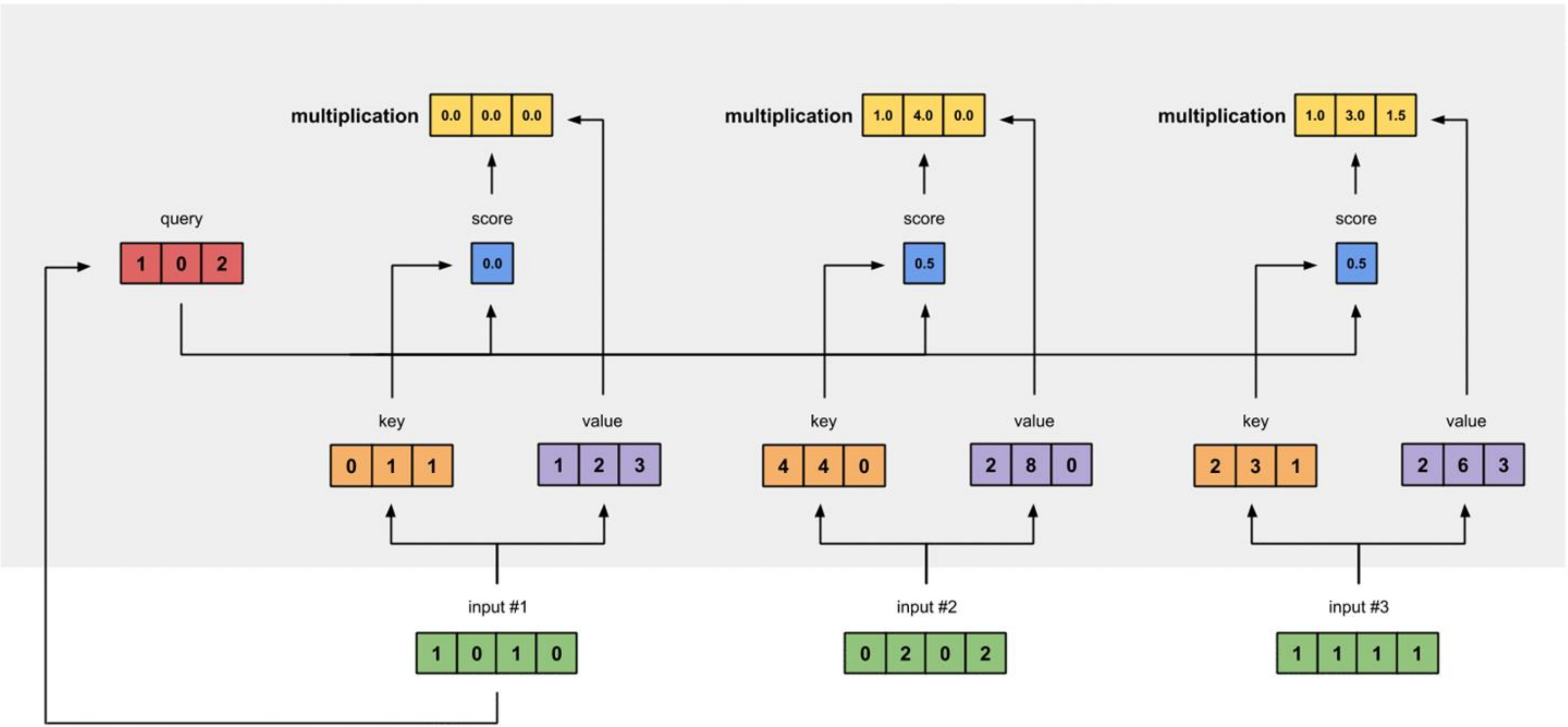
Self-attention

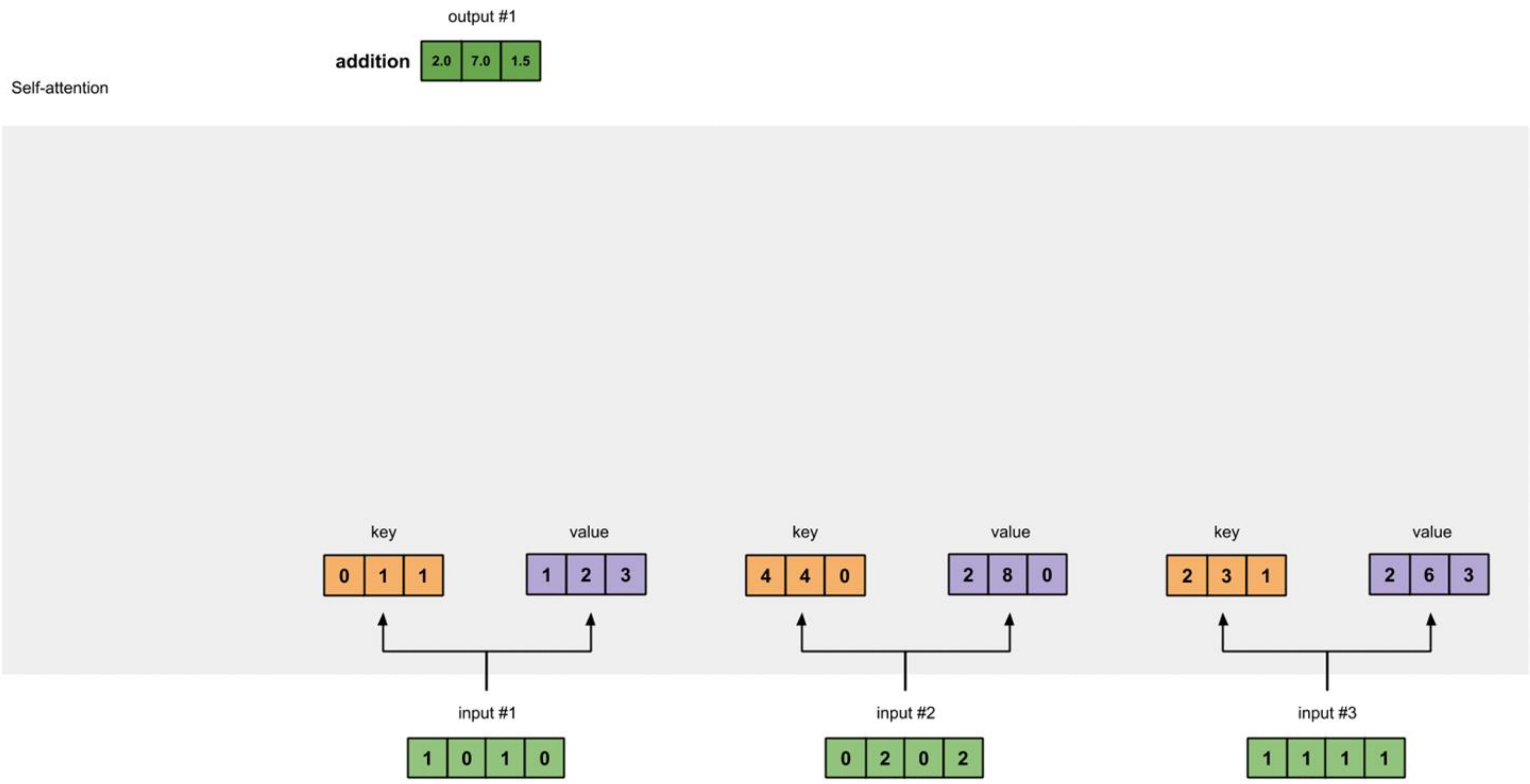


Self-attention



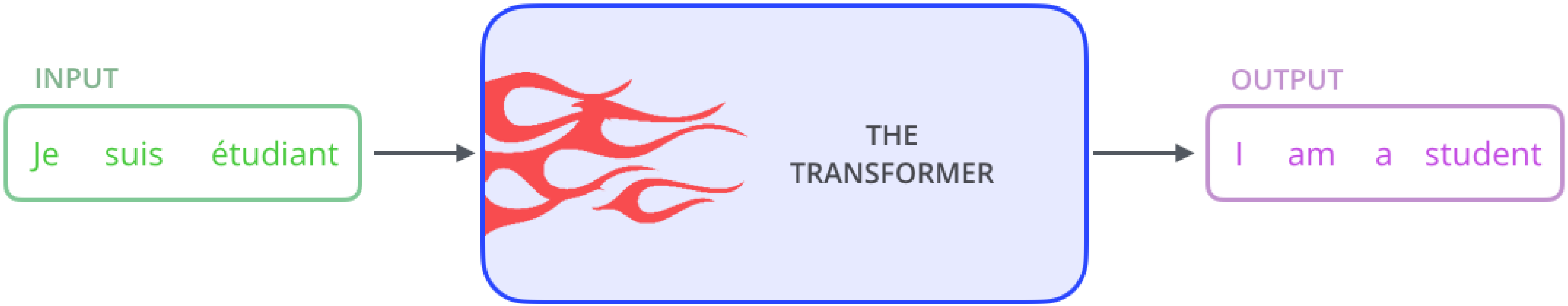
Self-attention



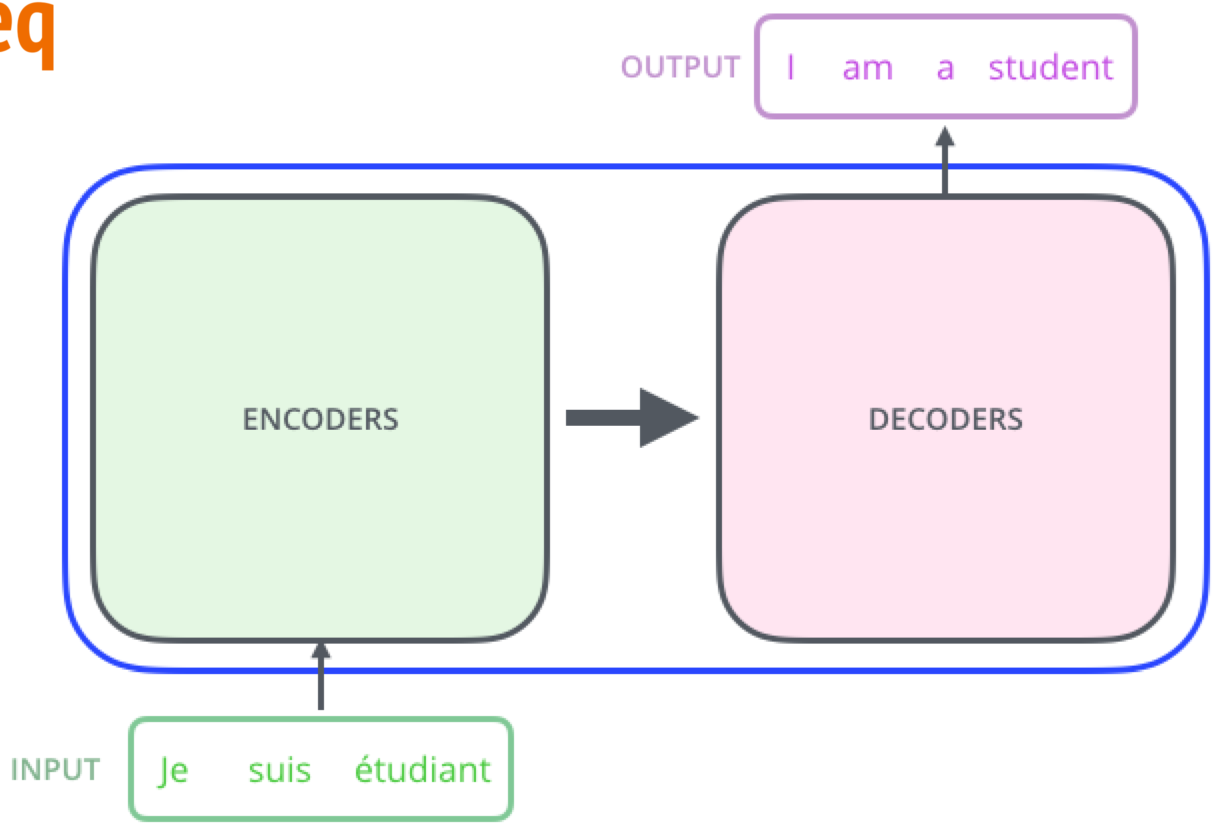


output #1
addition
2.0
7.0
1.5
Self-attention
key
0 1 1
value
1 2 3
key
4 4 0
value
2 8 0
key
2 3 1
value
2 6 3
input #1
1 0 1 0
input #2
0 2 0 2
input #3
1 1 1 1

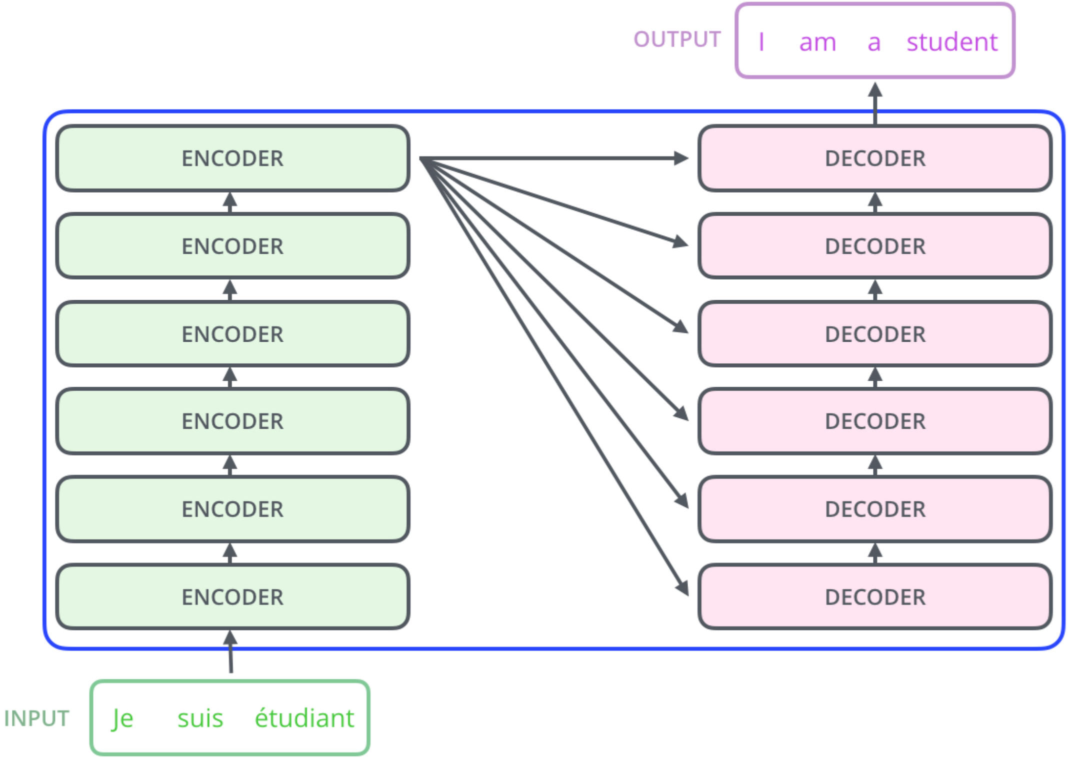
Transformer

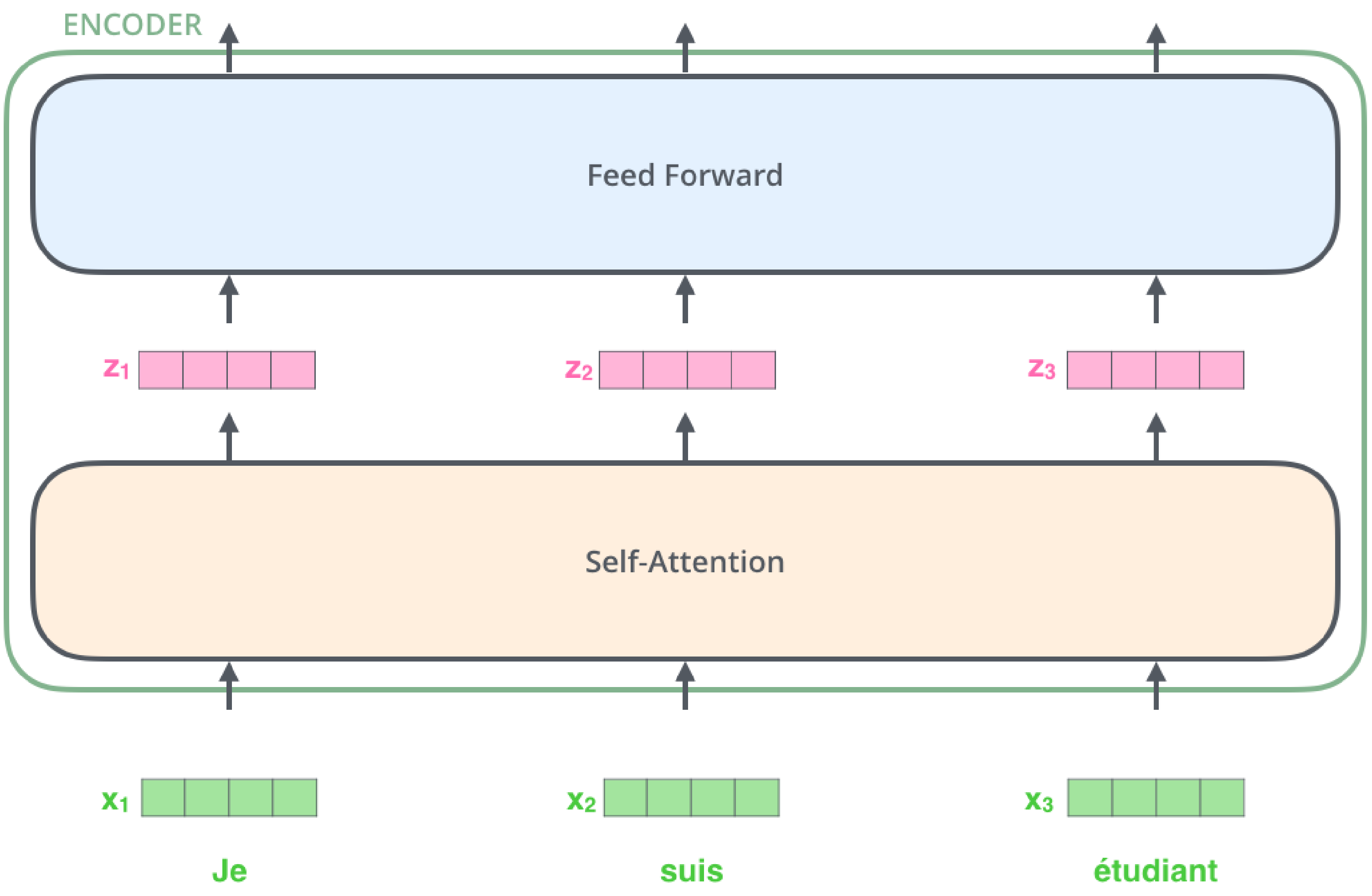


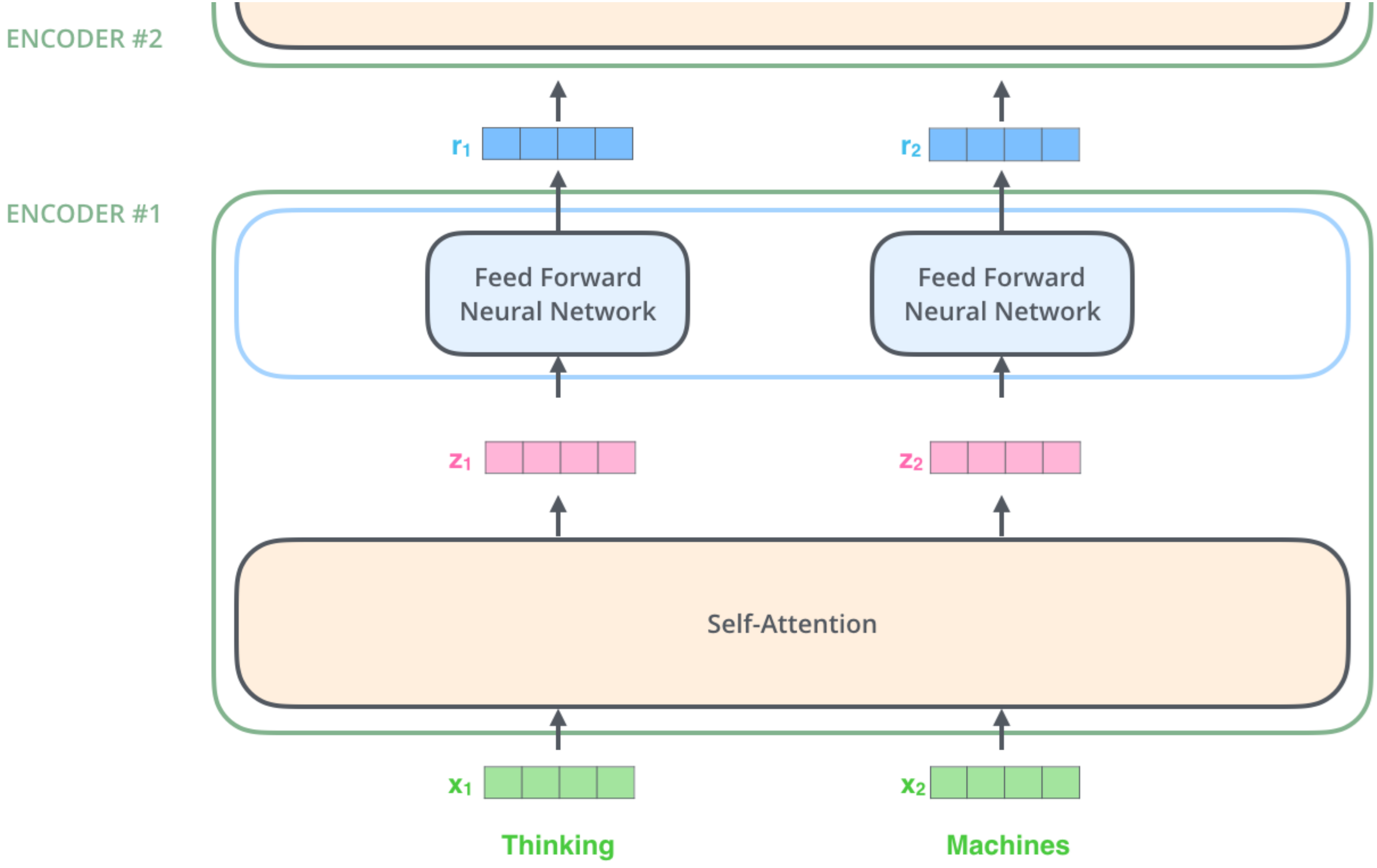
Seq2seq

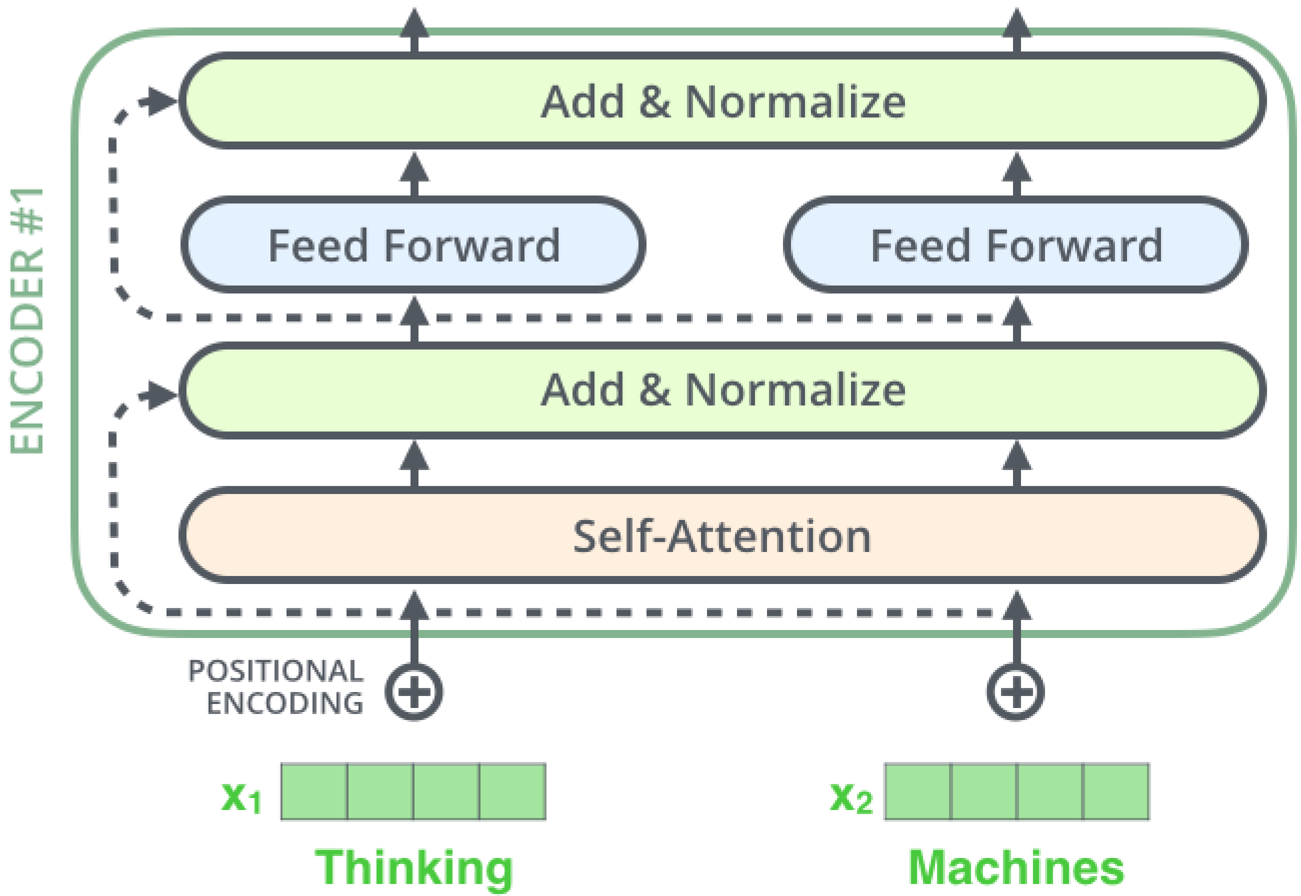

OUTPUT
I am a student
ENCODERS
DECODERS
INPUT
Je suis étudiant

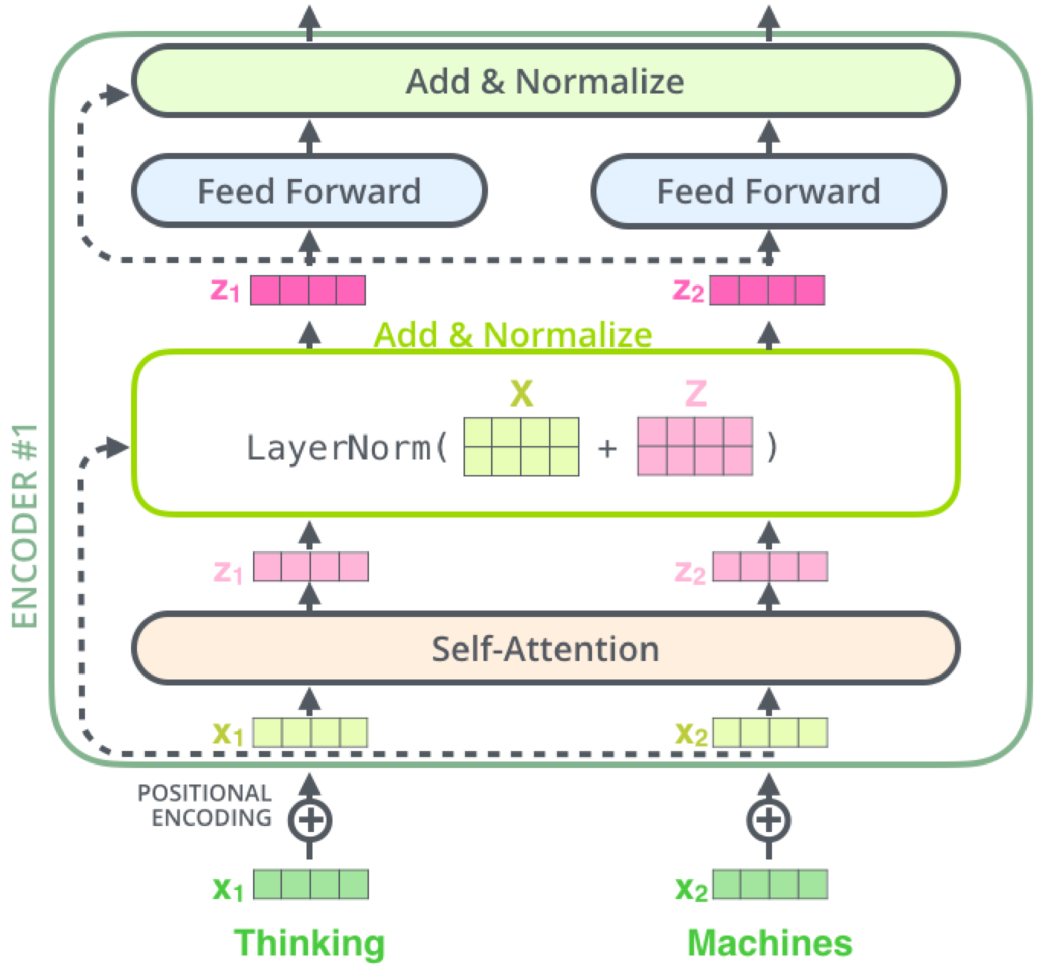
Transformer

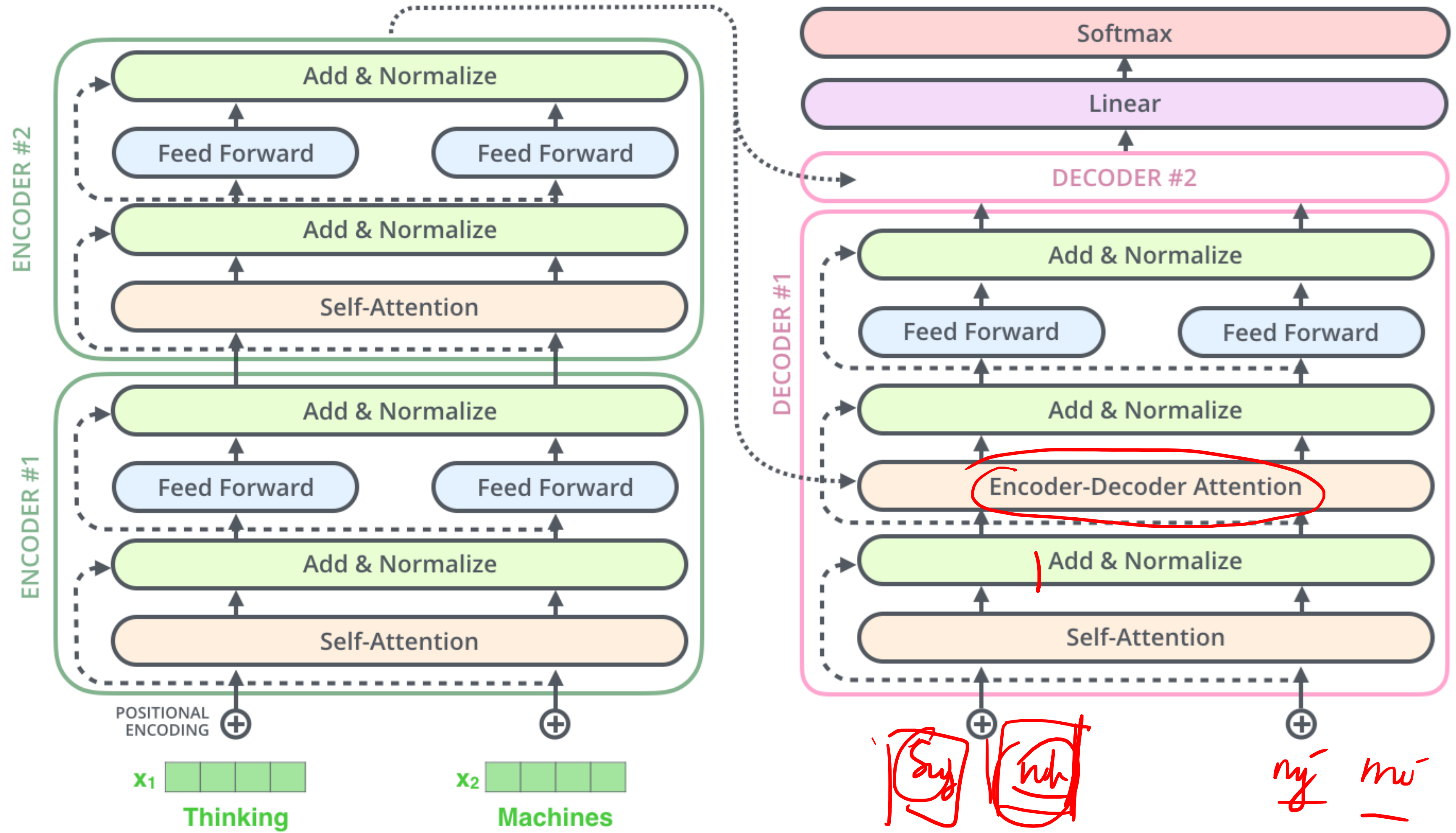


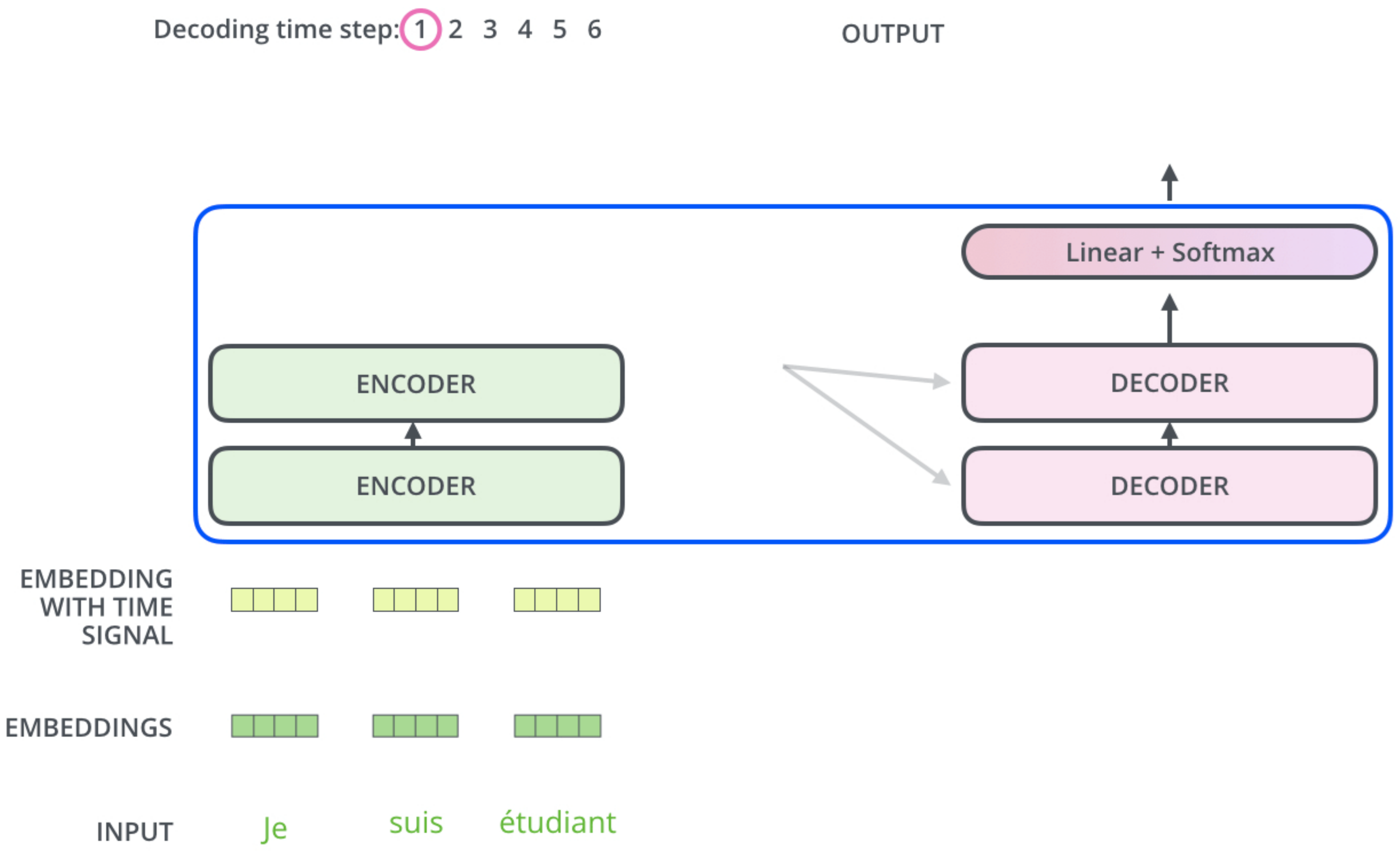

ENCODER
Feed Forward
z1
z2
z3
Self-Attention
x1
x2
x3
Je
suis
étudiant

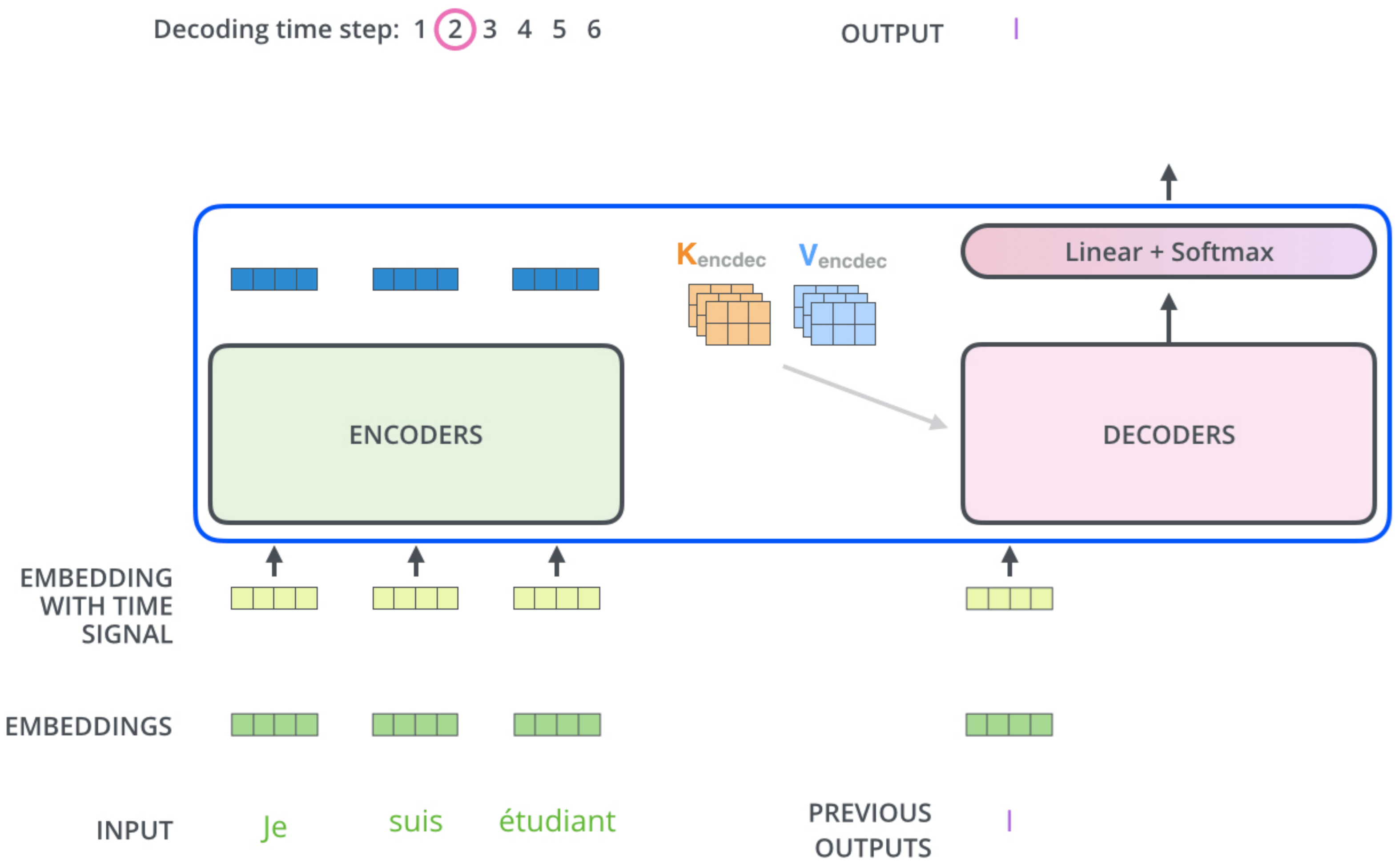

ENCODER #2
r1
r2
ENCODER #1
Feed Forward Neural Network
Feed Forward Neural Network
z1
z2
Self-Attention
x1
x2
Thinking
Machines


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

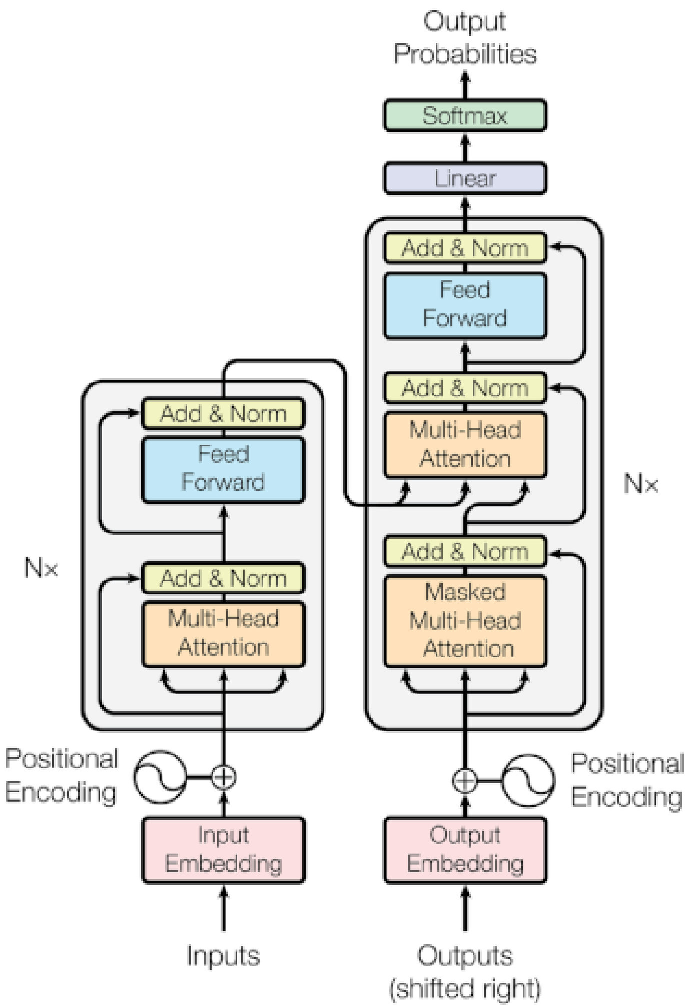

Add & Normalize
Feed Forward
Feed Forward
z1
z2
Add & Normalize
X
Z
LayerNorm(+)
z1
z2
Self-Attention
x1
x2
ENCODER #1
POSITIONAL
ENCODING
x1
x2
Thinking
Machines


Softmax
Linear
DECODER #2
DECODER #1
Add & Normalize
Feed Forward
Feed Forward
Add & Normalize
Encoder-Decoder Attention
Add & Normalize
Self-Attention
ENCODER #2
Add & Normalize
Feed Forward
Feed Forward
Add & Normalize
Self-Attention
ENCODER #1
Add & Normalize
Feed Forward
Feed Forward
Add & Normalize
Self-Attention
POSITIONAL ENCODING
x_1
Thinking
x_2
Machines


Decoding time step: 1 2 3 4 5 6
OUTPUT
Linear + Softmax
ENCODER
ENCODER
DECODER
DECODER
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant


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

Transformer



Multi-head attention

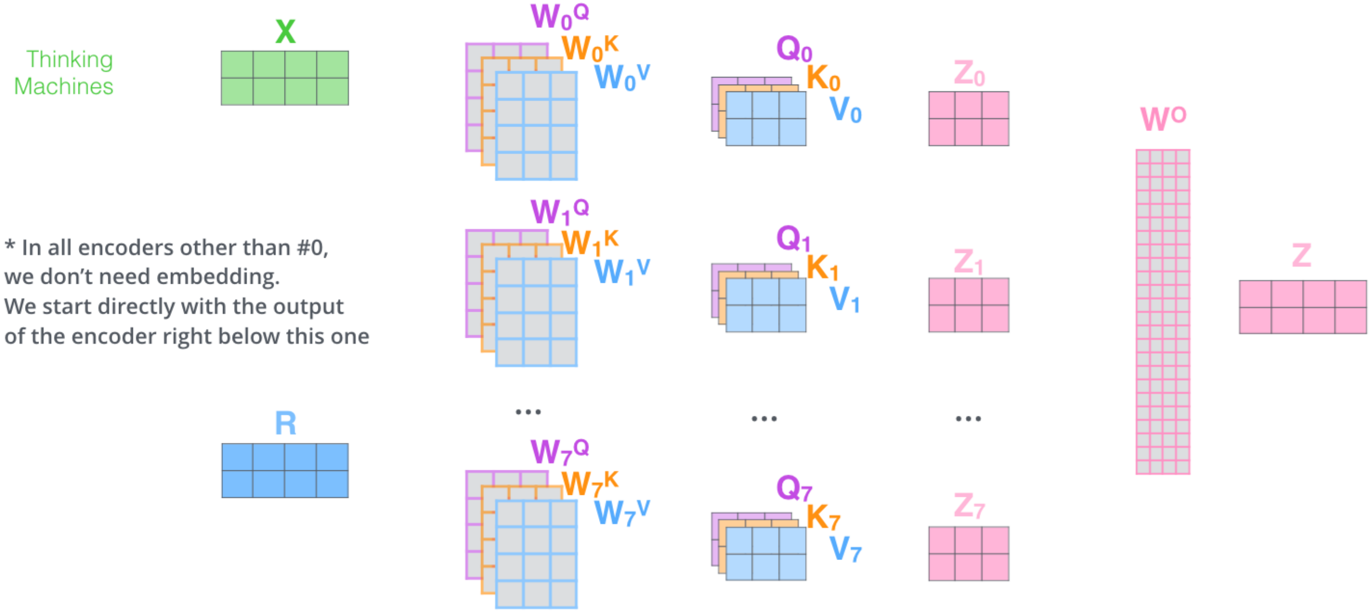
1) This is our input sentence*

2) We embed each word*

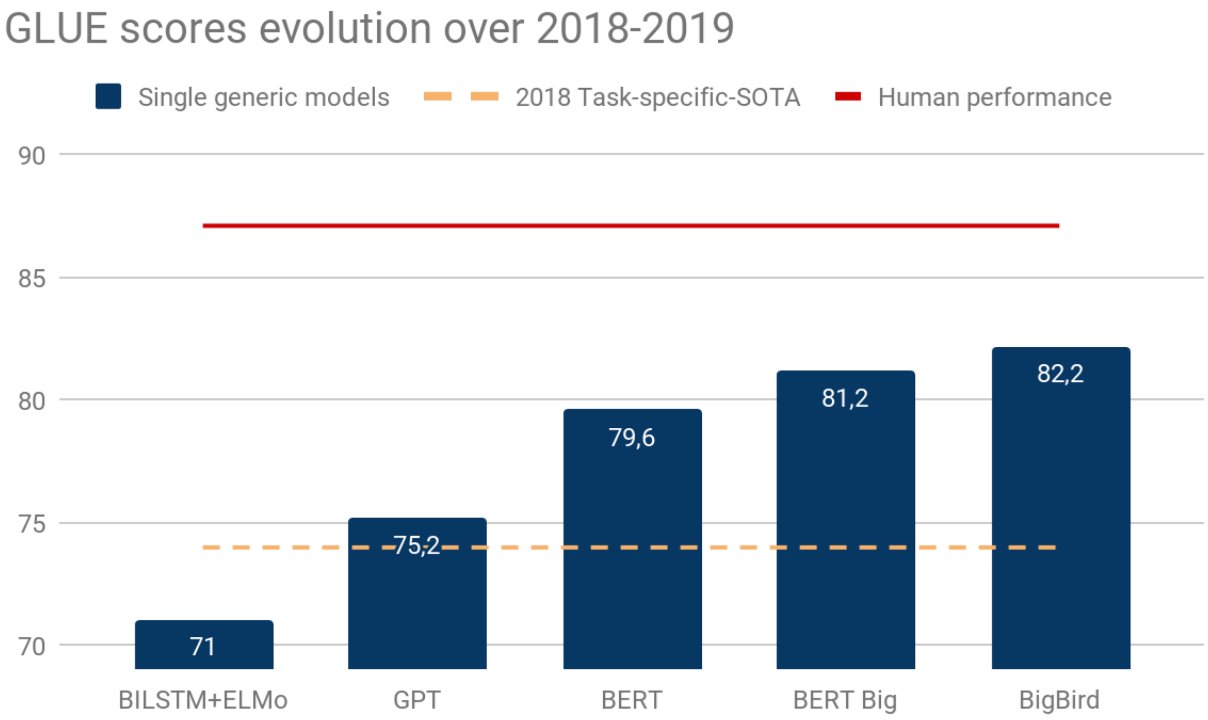
3) Split into 8 heads. We multiply X or R with weight matrices

4) Calculate attention using the resulting Q/K/V matrices

5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer

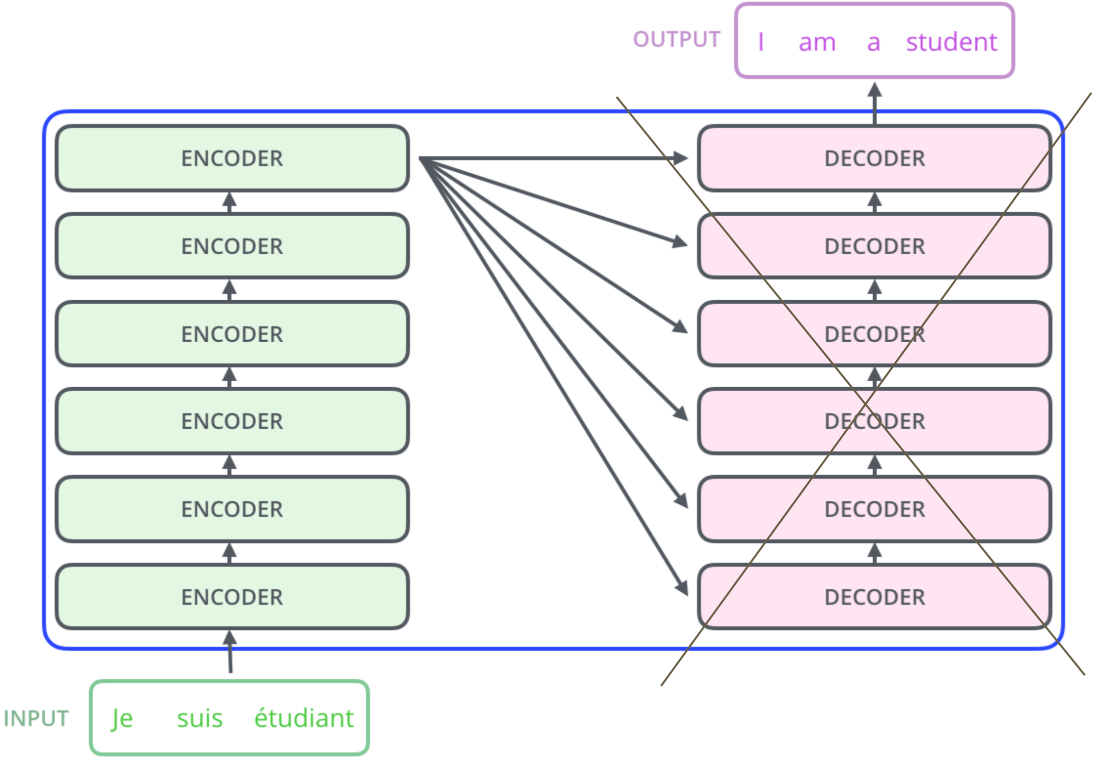


BERT



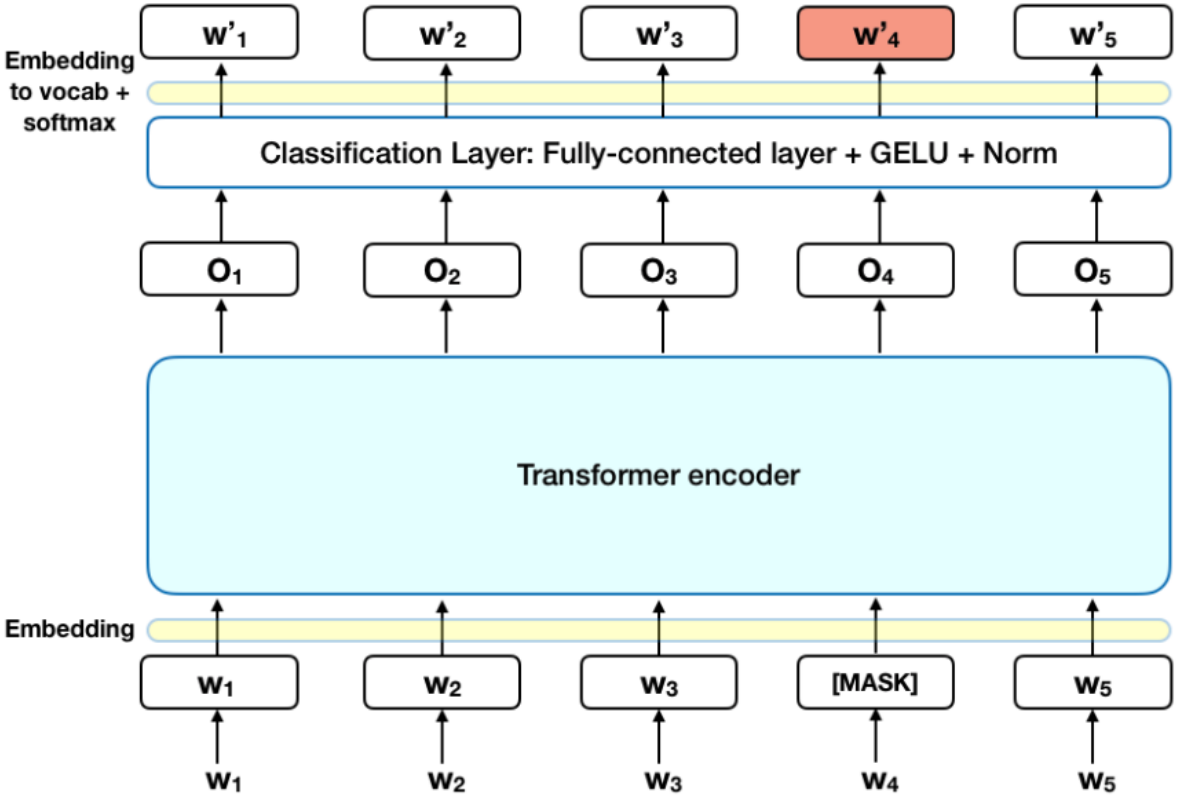
GLUE scores evolution over 2018-2019
Single generic models
2018 Task-specific-SOTA
Human performance
90
85
80
75
70
71
75,2
79,6
81,2
82,2
BILSTM+ELMo
GPT
BERT
BERT Big
BigBird

BERT model

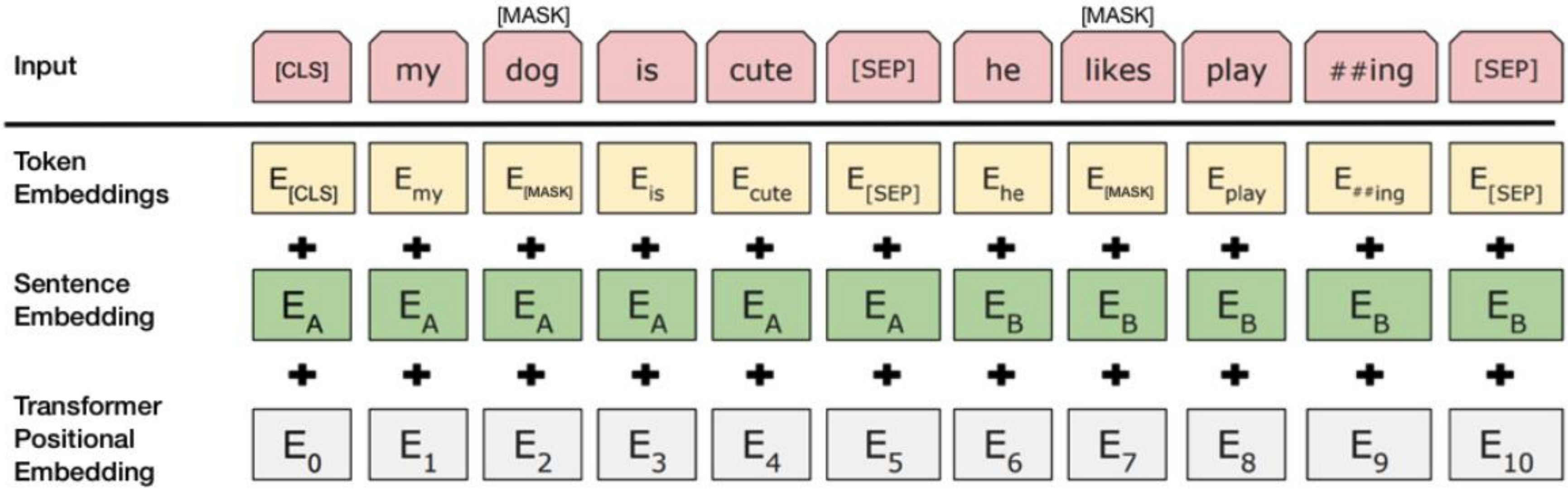


- Mask Language Modeling (MLM)
- Next Sentence Prediction (NSP)

Masked LM (MLM)



Next sentence prediction



Fine-tuning BERT

- Classification tasks such as sentiment analysis.
- In Question Answering tasks (e.g. SQuAD v1.1).
- In Named Entity Recognition (NER).

Q&A


@LateNightSeth
THANK YOU!