Deep Learning

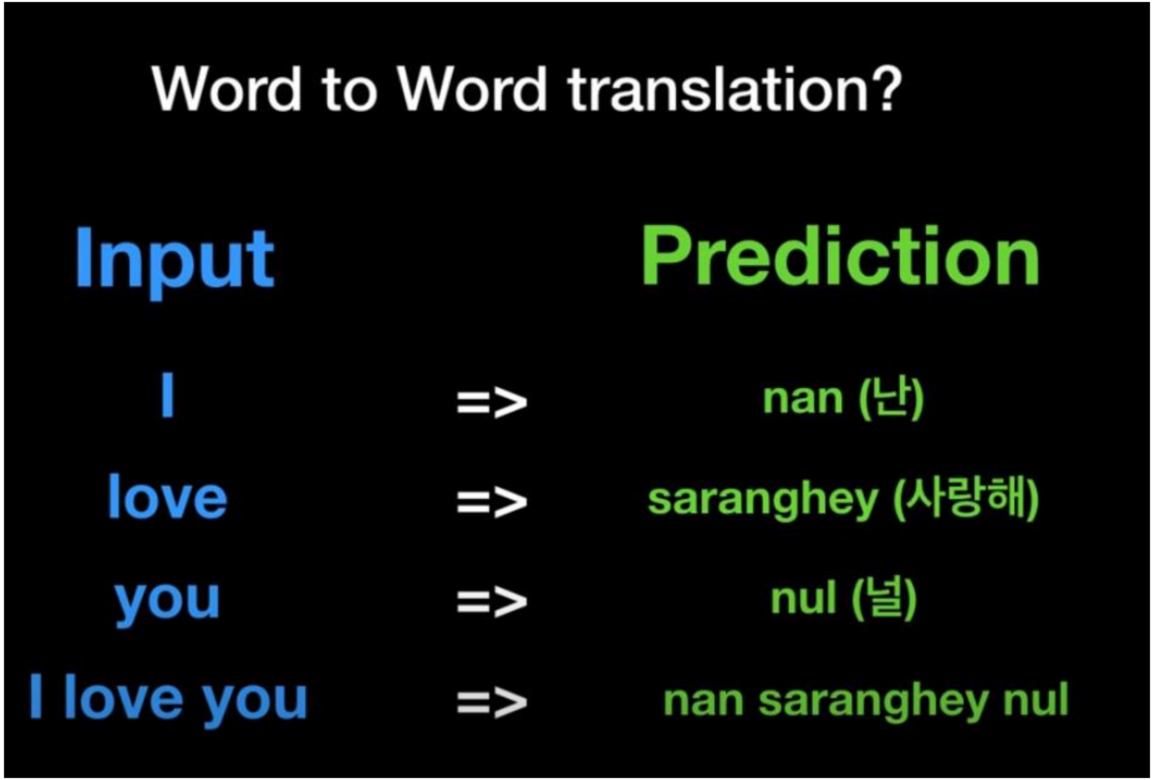
6주차

12기 이두형
12기 임효진

KU-BIG

seq2seq

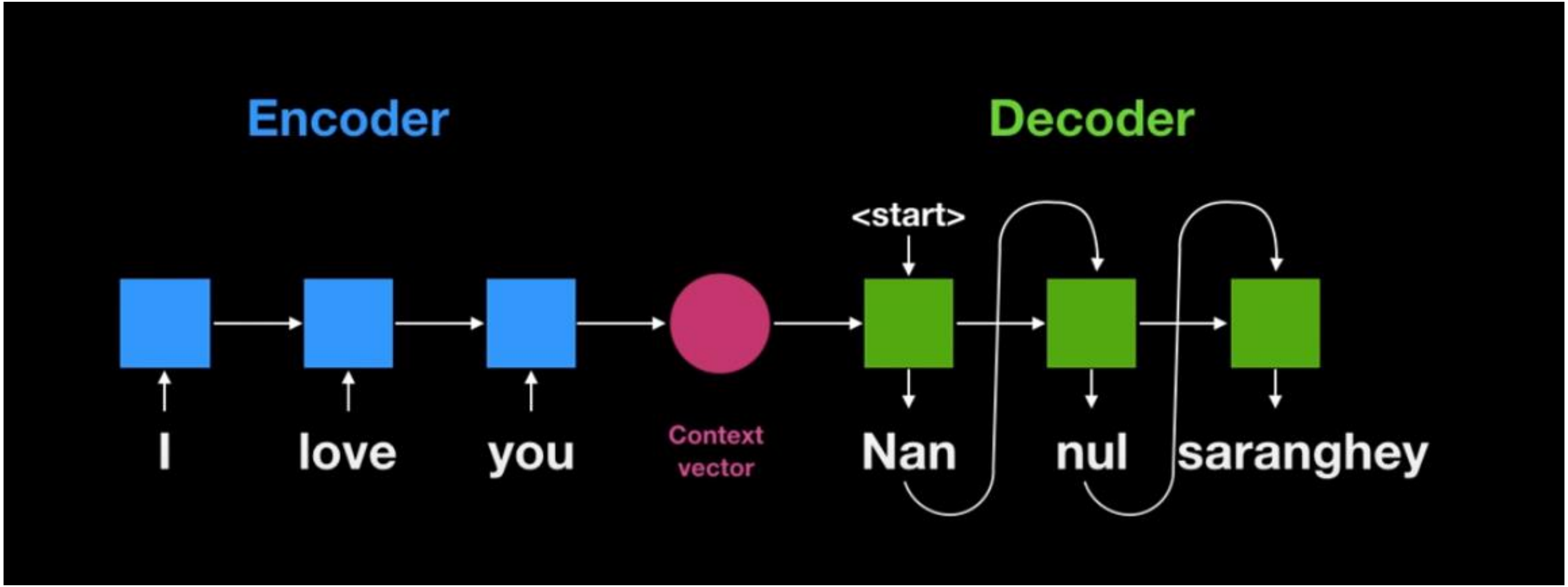

What is the best way for translation?
I love you = Nan nul saranghey
난 널 사랑해


Word to Word translation?
Input
Prediction
I => nan (난)
love => saranghey (사랑해)
you => nul (널)
I love you => nan saranghey nul

How are you ? = Jal jiney ?

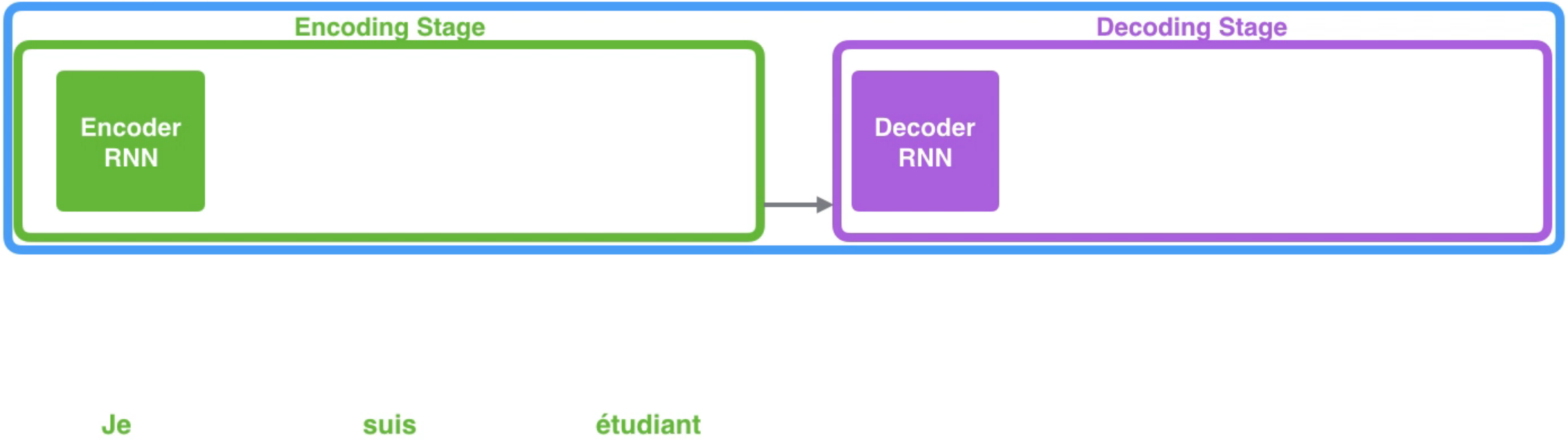
잘 지내

3 words 2 words


Encoder
Decoder
<start>
I
love
you
Context vector
Nan
nul
saranghey

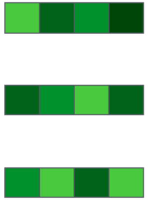
Neural Machine Translation

SEQUENCE TO SEQUENCE MODEL



Input

Je	0.901	-0.651	-0.194	-0.822
suis	-0.351	0.123	0.435	-0.200
étudiant	0.081	0.458	-0.400	0.480



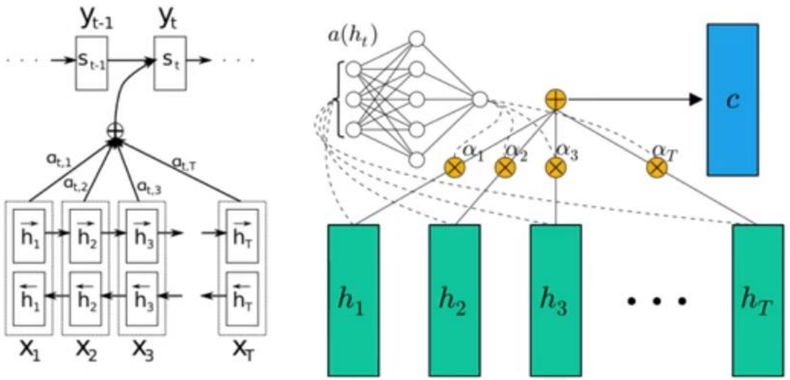
CONTEXT

0.11
0.03
0.81
-0.62

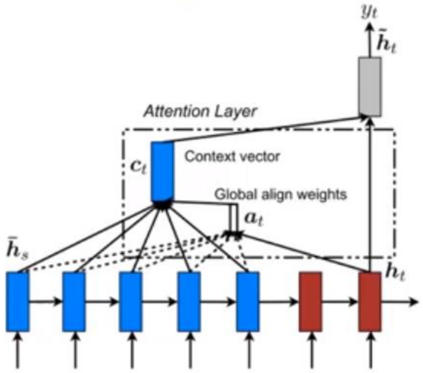
0.11
0.03
0.81
-0.62

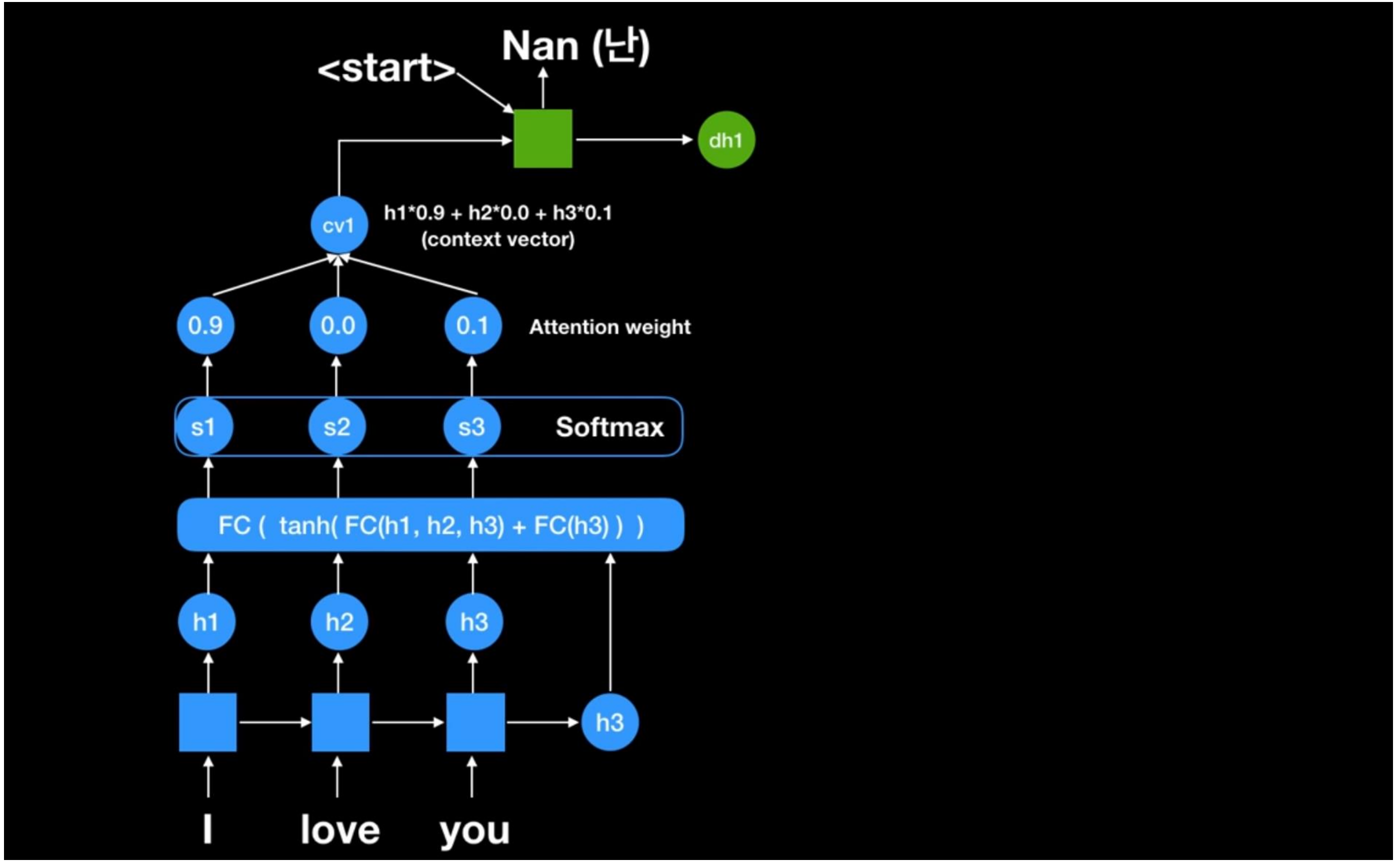
- Attention
  - ✓ Bahadanau attention (Bahdanau et al., 2015)
    - Attention scores are separated trained, the current hidden state is a function of the context vector and the previous hidden state
  - ✓ Luong attention (Luong et al., 2015)
    - Attention scores are not trained, the new current hidden state is the simple tanh of the weigthed concatenation of the context vector and the current hidden state of the decoder

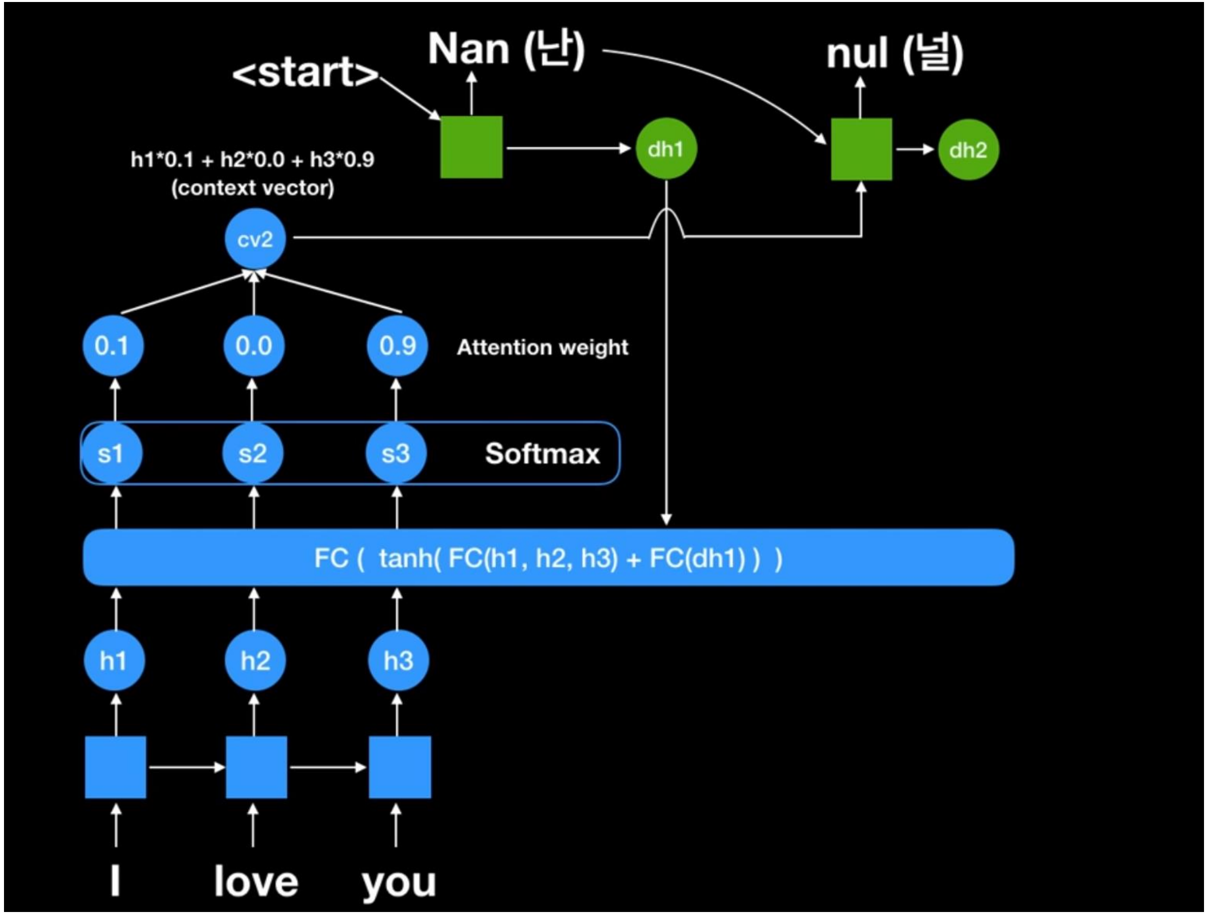
Bahdanau attention

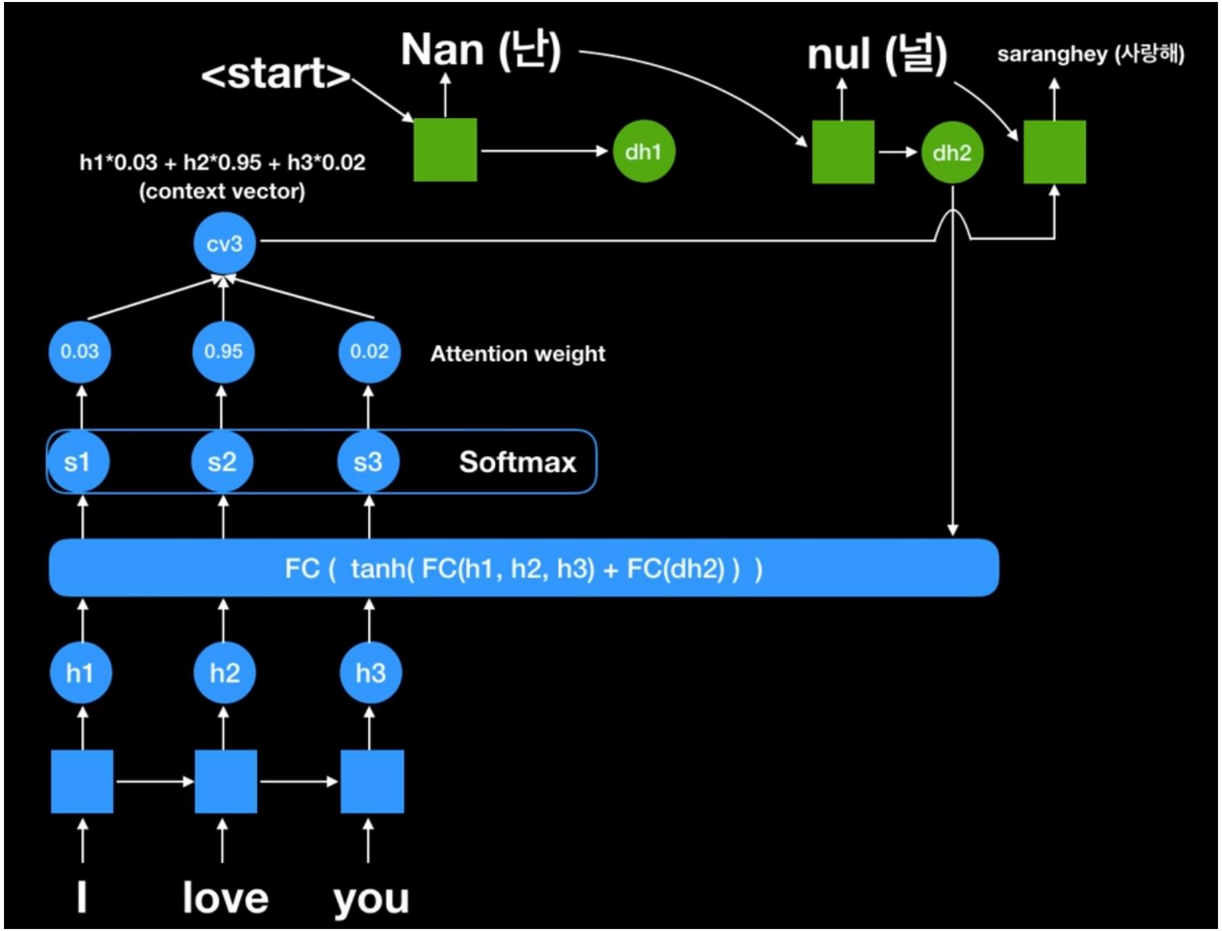


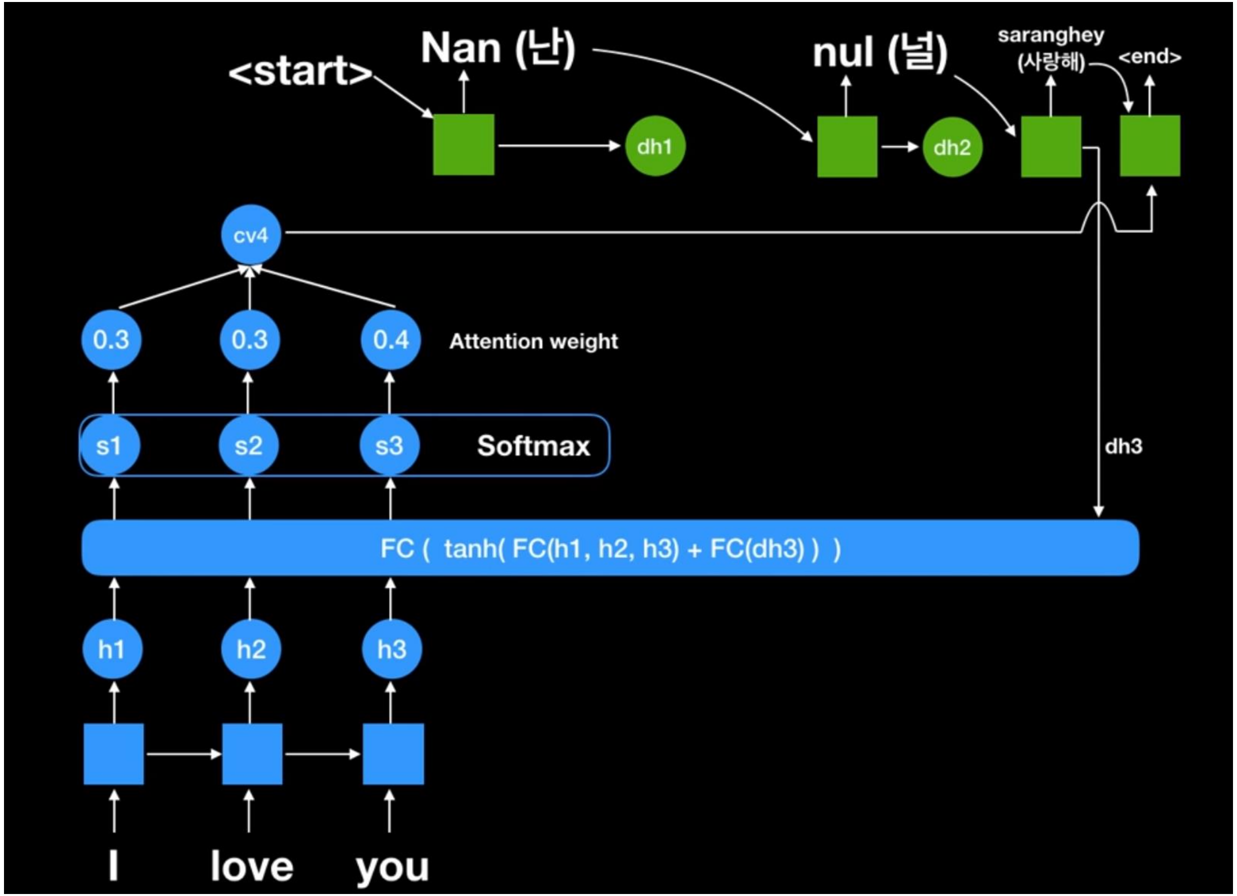
Luong attention




<start>
Nan (난)
dh1
cv1
h1*0.9 + h2*0.0 + h3*0.1
(context vector)
0.9
0.0
0.1
Attention weight
s1
s2
s3
Softmax
FC (tanh(FC(h1, h2, h3) + FC(h3)))
h1
h2
h3
h3
I
love
you


<start>
Nan (난)
nul (널)
dh1
dh2
h1*0.1 + h2*0.0 + h3*0.9
(context vector)
cv2
0.1
0.0
0.9
Attention weight
s1
s2
s3
Softmax
FC (tanh(FC(h1, h2, h3) + FC(dh1)))
h1
h2
h3
I
love
you

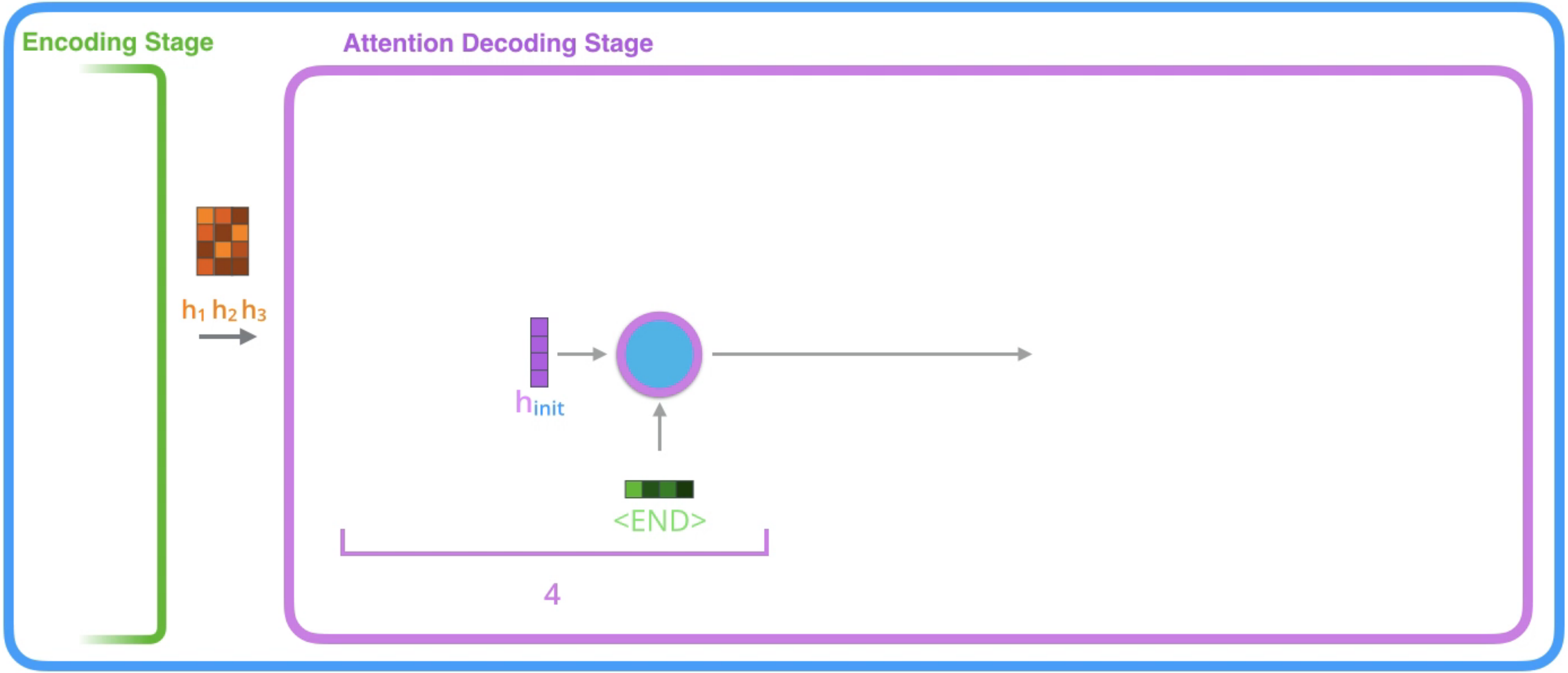

<start>
Nan (난)
nul (널)
saranghey (사랑해)
dh1
dh2
h1*0.03 + h2*0.95 + h3*0.02
(context vector)
cv3
0.03
0.95
0.02
Attention weight
s1
s2
s3
Softmax
FC (tanh(FC(h1, h2, h3) + FC(dh2)))
h1
h2
h3
I
love
you

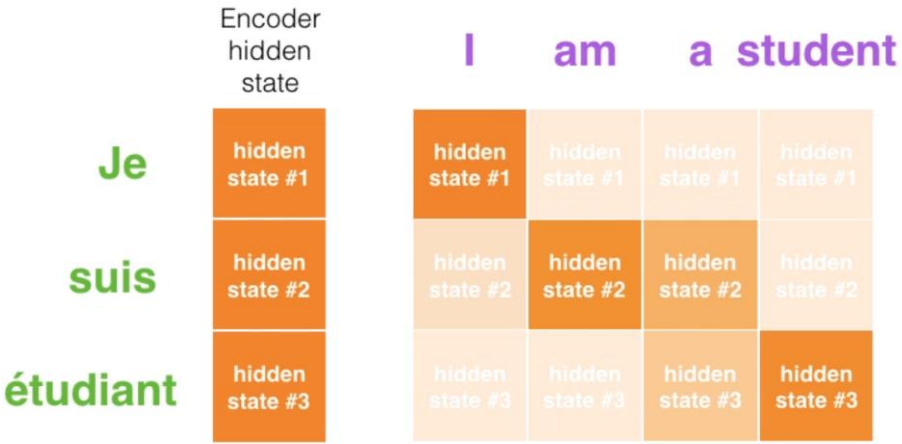

<start>
Nan (난)
nul (널)
saranghey
(사랑해)
<end>
dh1
dh2
cv4
0.3
0.3
0.4
Attention weight
s1
s2
s3
Softmax
dh3
FC (tanh(FC(h1, h2, h3) + FC(dh3)))
h1
h2
h3
I
love
you

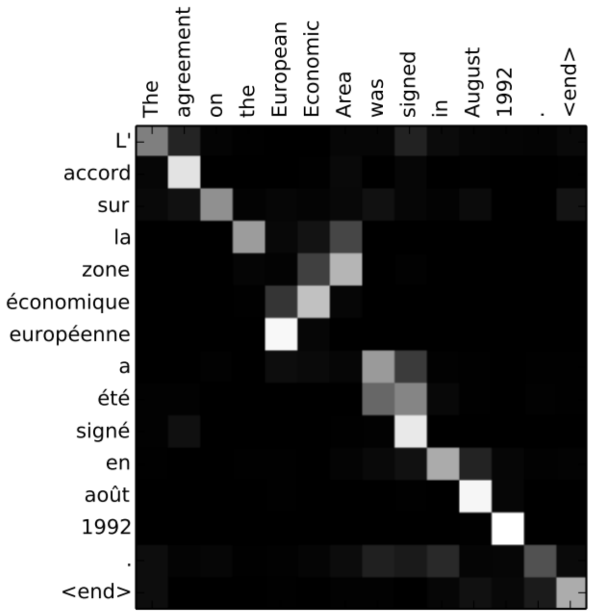
Attention at time step 4

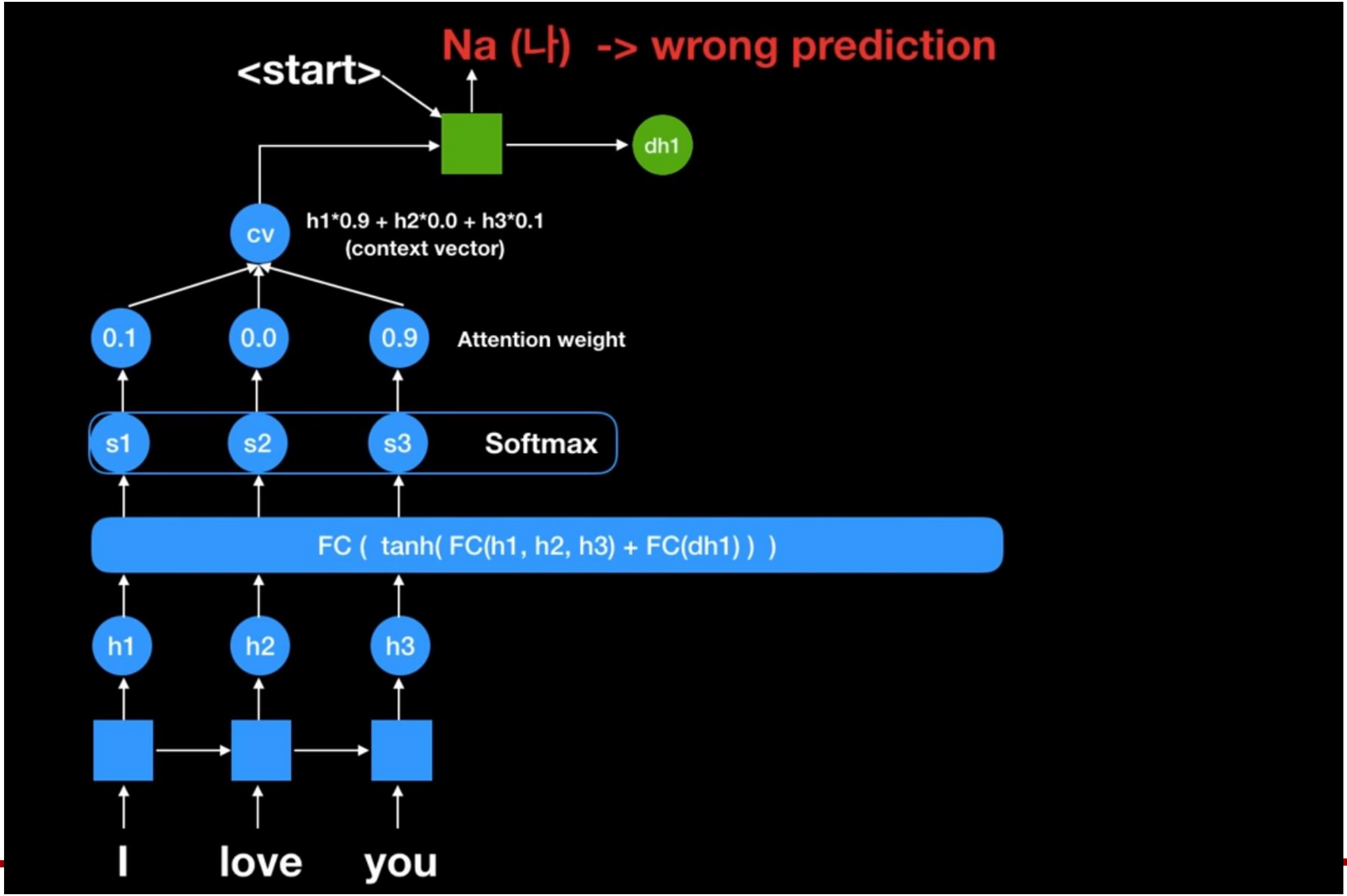
Neural Machine Translation

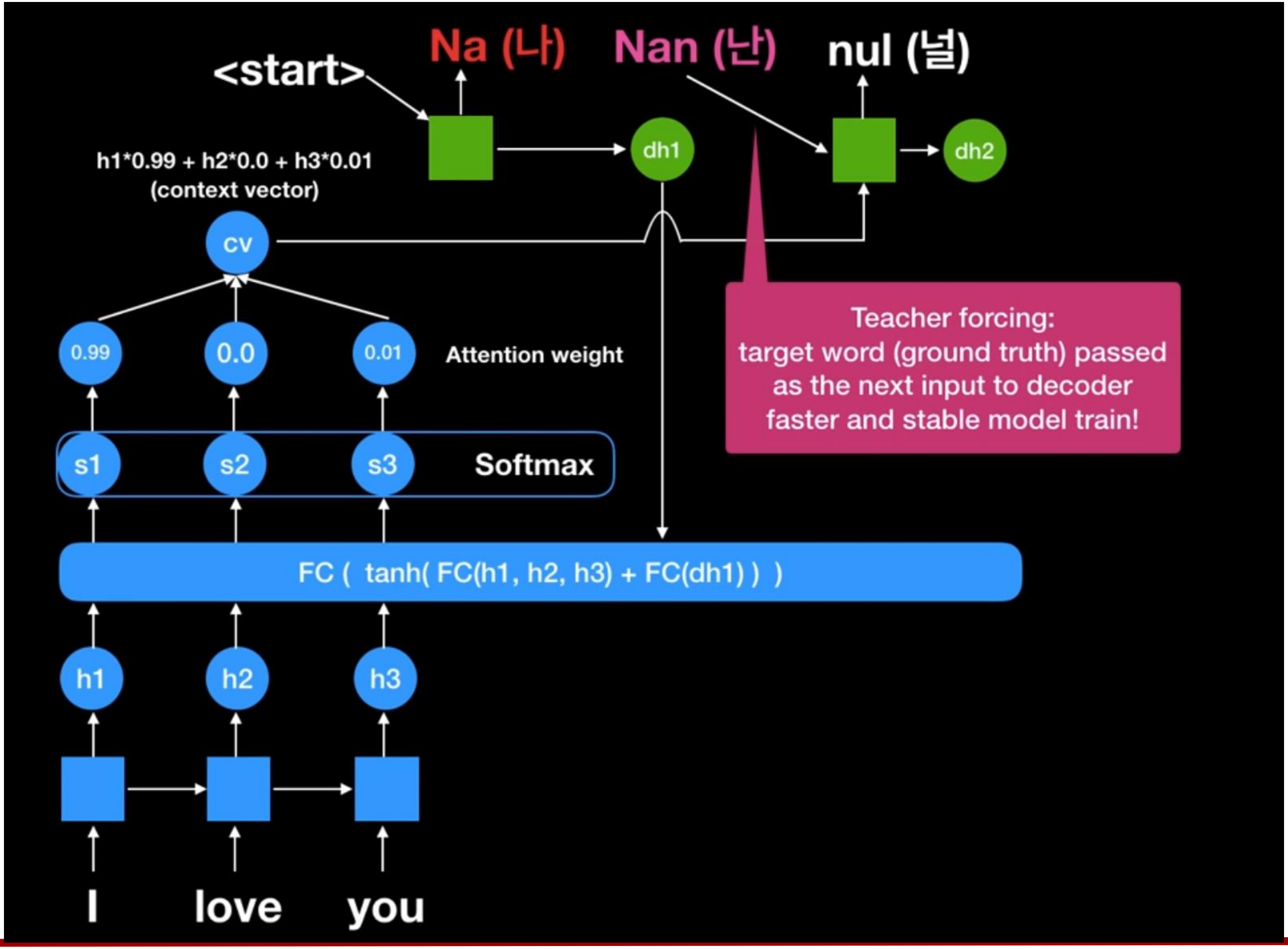
SEQUENCE TO SEQUENCE MODEL WITH ATTENTION

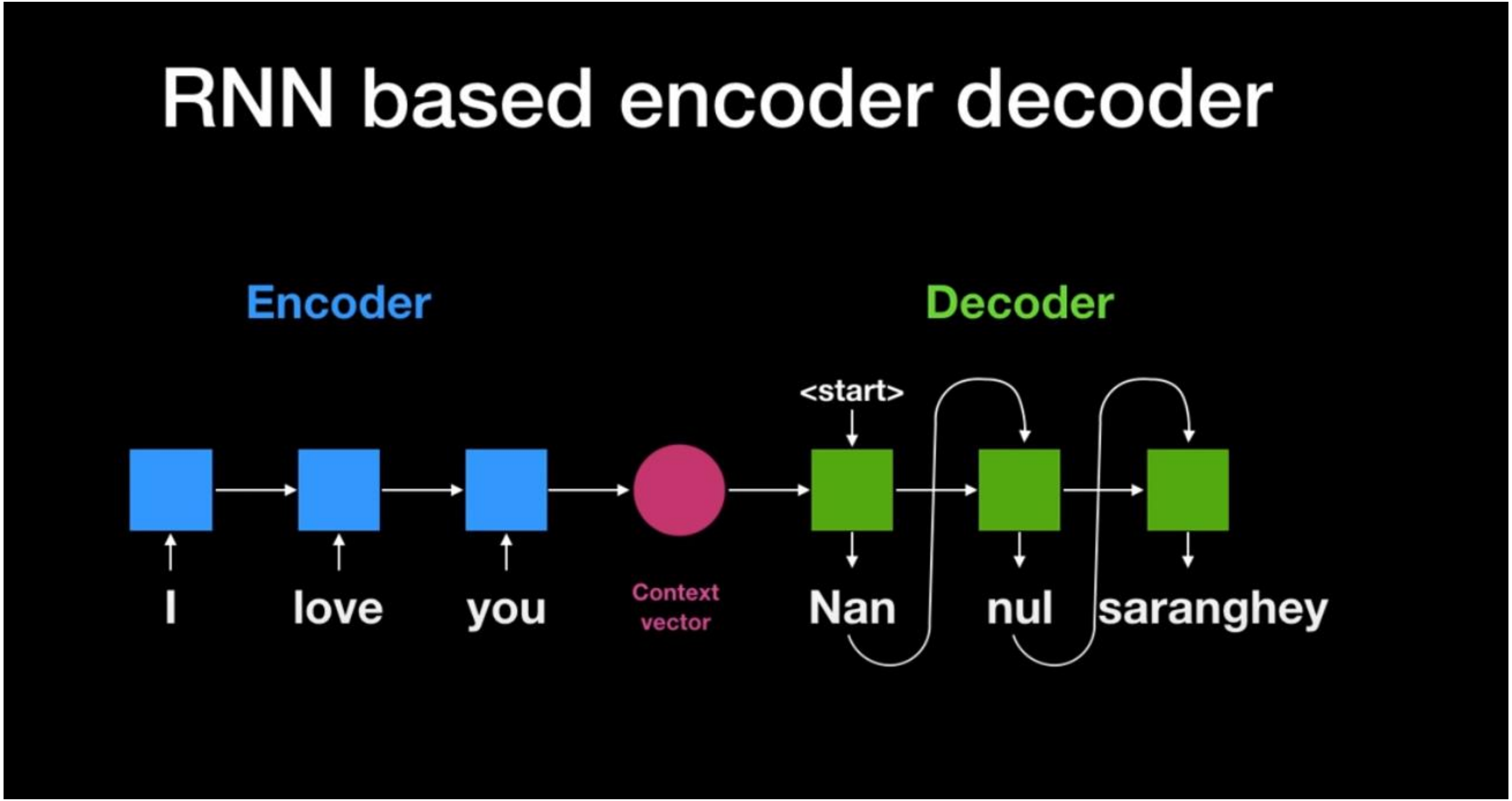


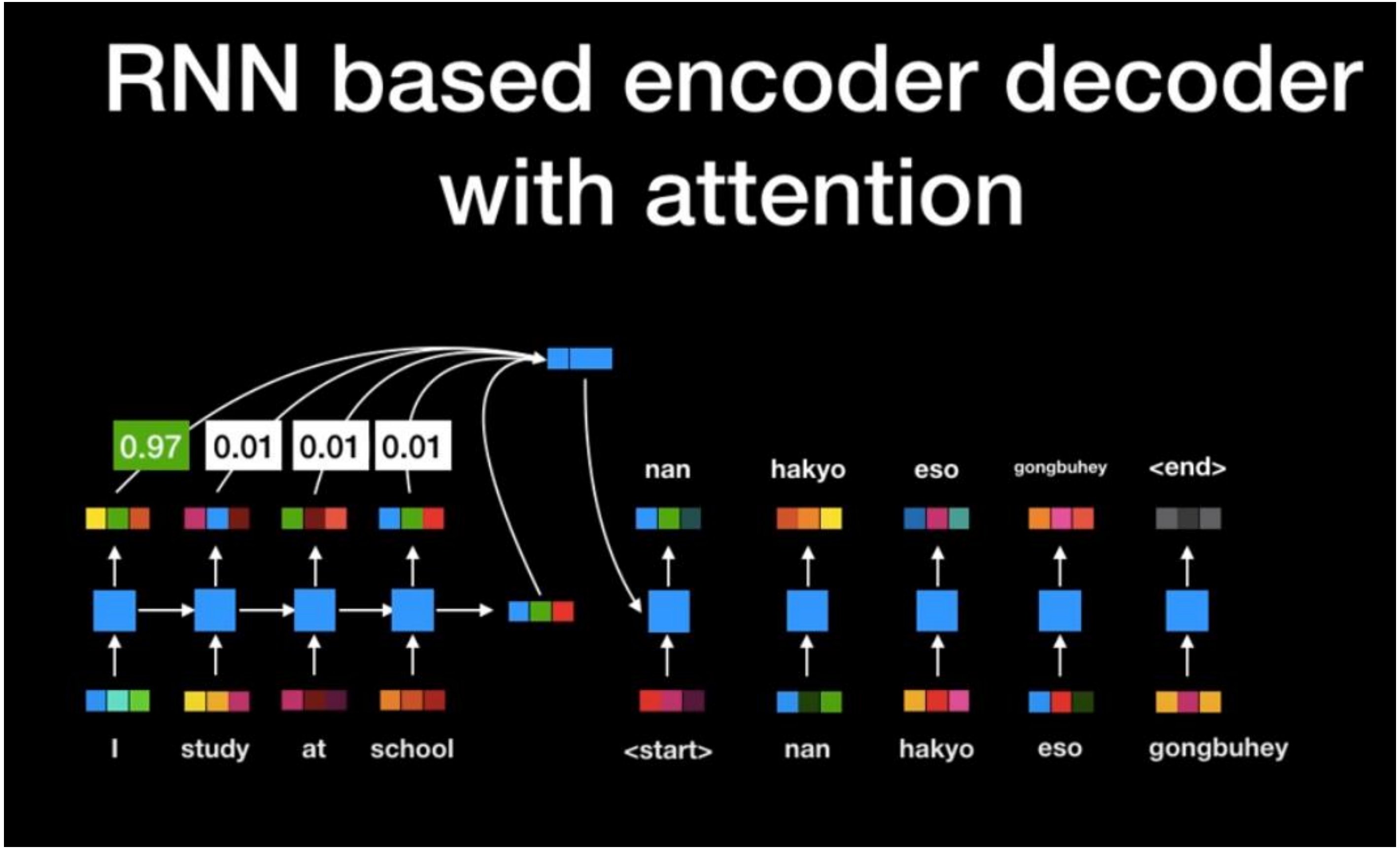

Encoder hidden state
I am a student
Je
suis
étudiant
hidden state #1
hidden state #2
hidden state #3
hidden state #1
hidden state #1
hidden state #1
hidden state #1
hidden state #2
hidden state #2
hidden state #2
hidden state #2
hidden state #3
hidden state #3
hidden state #3
hidden state #3

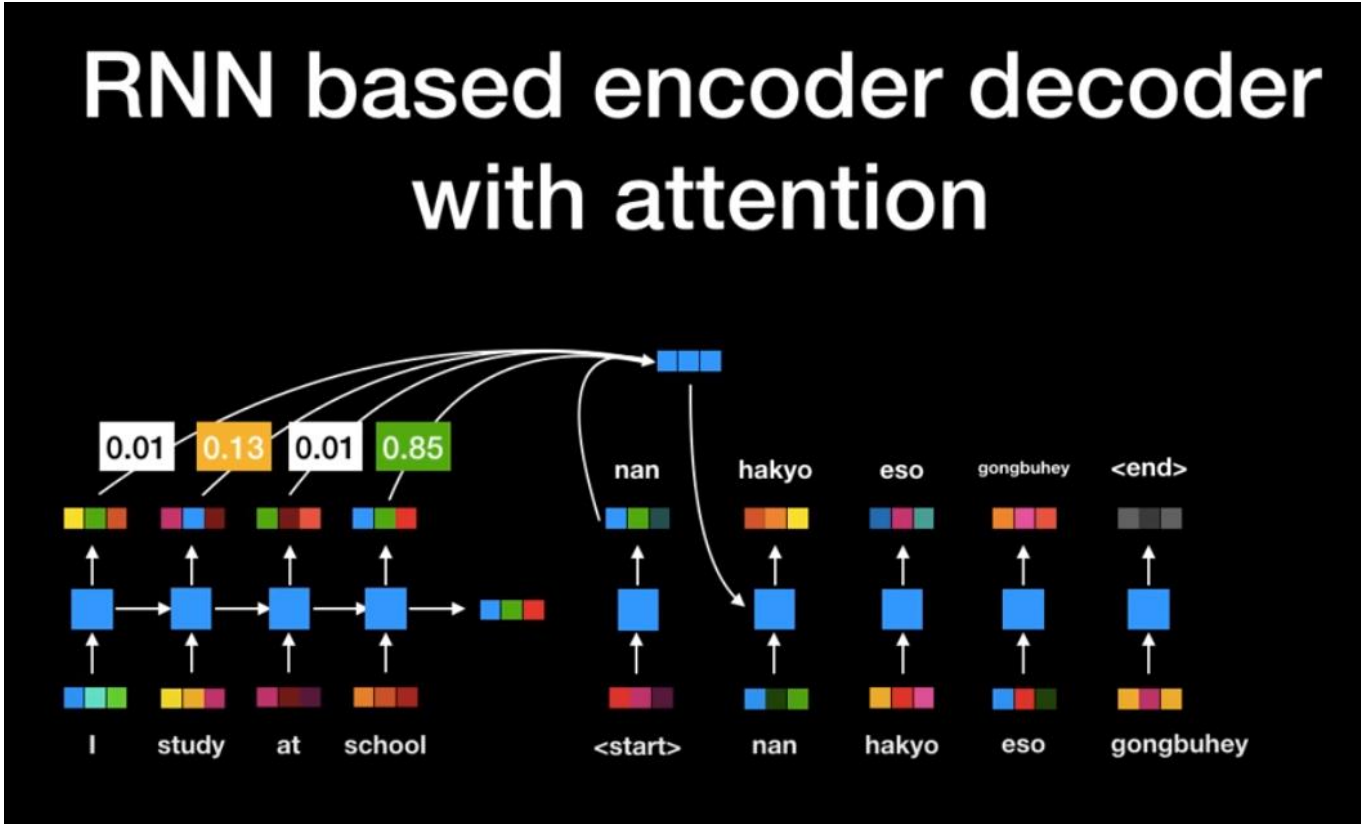

The
agreement
on
the
European
Economic
Area
was
signed
in
August
1992
.
<end>
L'
accord
sur
la
zone
économique
européenne
a
été
signé
en
août
1992
.
<end>


Na (나) -> wrong prediction
<start>
dh1
cv
h1*0.9 + h2*0.0 + h3*0.1
(context vector)
0.1
0.0
0.9
Attention weight
s1
s2
s3
Softmax
FC (tanh(FC(h1, h2, h3) + FC(dh1)))
h1
h2
h3
I
love
you

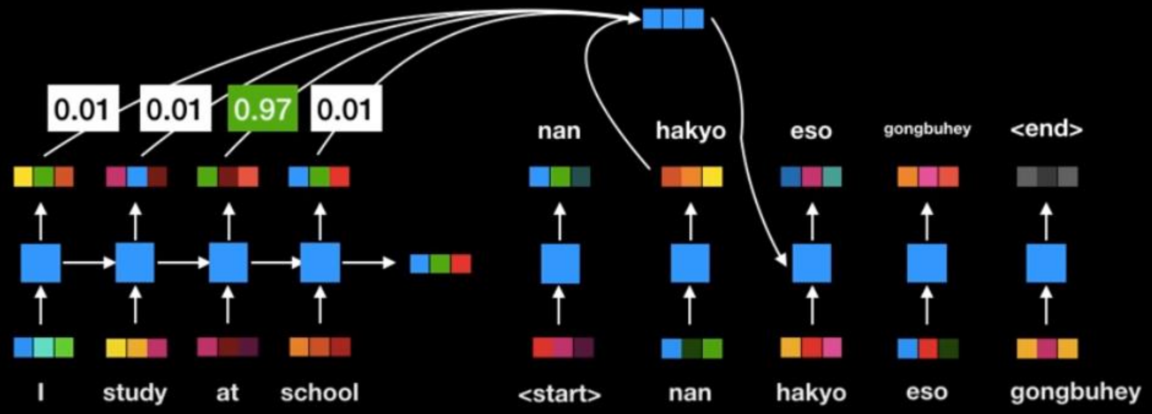

<start>
Na (나)
Nan (난)
nul (널)
h1*0.99 + h2*0.0 + h3*0.01
(context vector)
cv
dh1
dh2
0.99
0.0
0.01
Attention weight
Teacher forcing:
target word (ground truth) passed
as the next input to decoder
faster and stable model train!
s1
s2
s3
Softmax
FC (tanh(FC(h1, h2, h3) + FC(dh1)))
h1
h2
h3
I
love
you


RNN based encoder decoder
Encoder
Decoder
<start>
I
love
you
Context vector
Nan
nul
saranghey

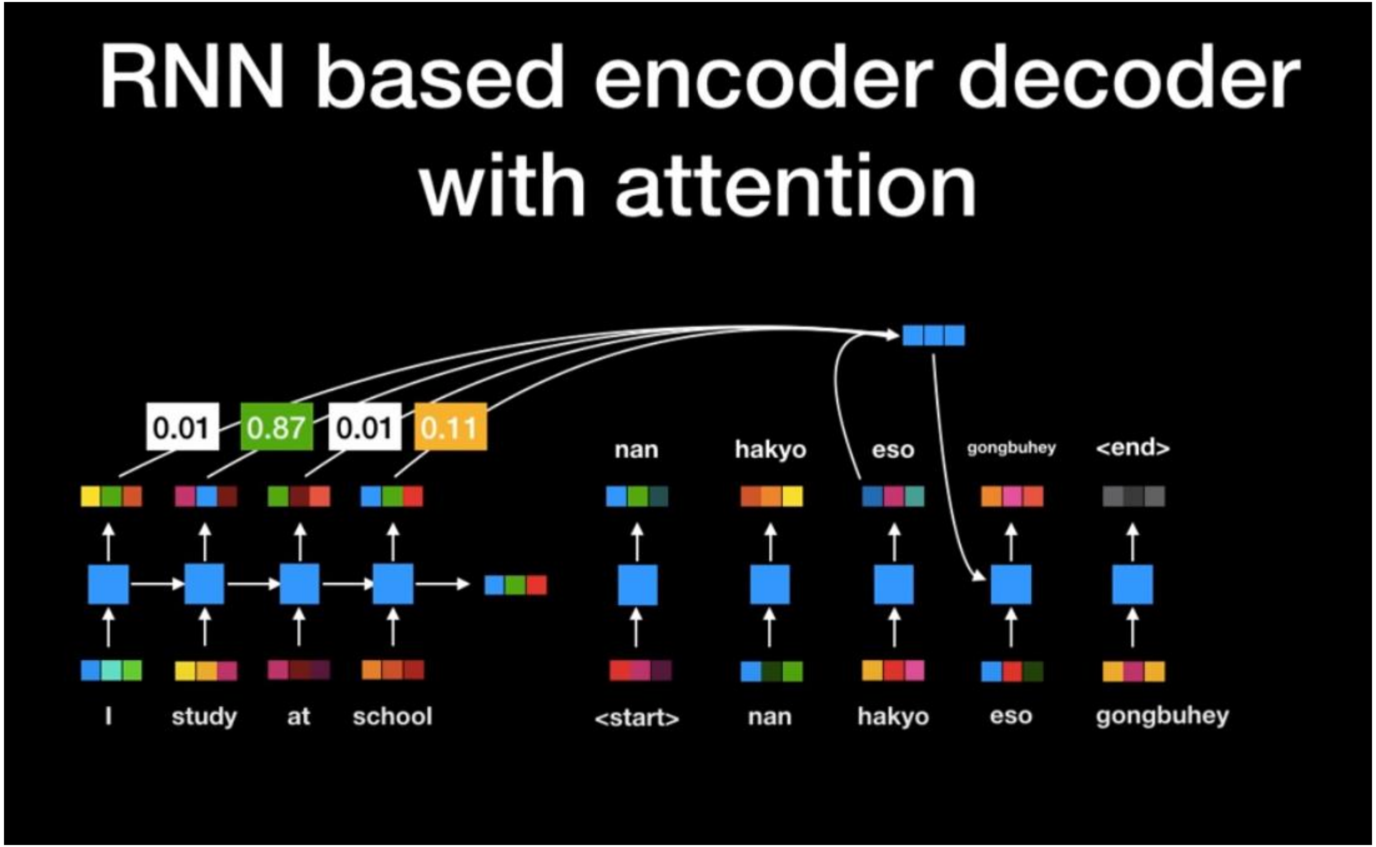

RNN based encoder decoder with attention
0.97
0.01
0.01
0.01
nan
hakyo
eso
gongbuhey
<end>
I
study
at
school
<start>
nan
hakyo
eso
gongbuhey

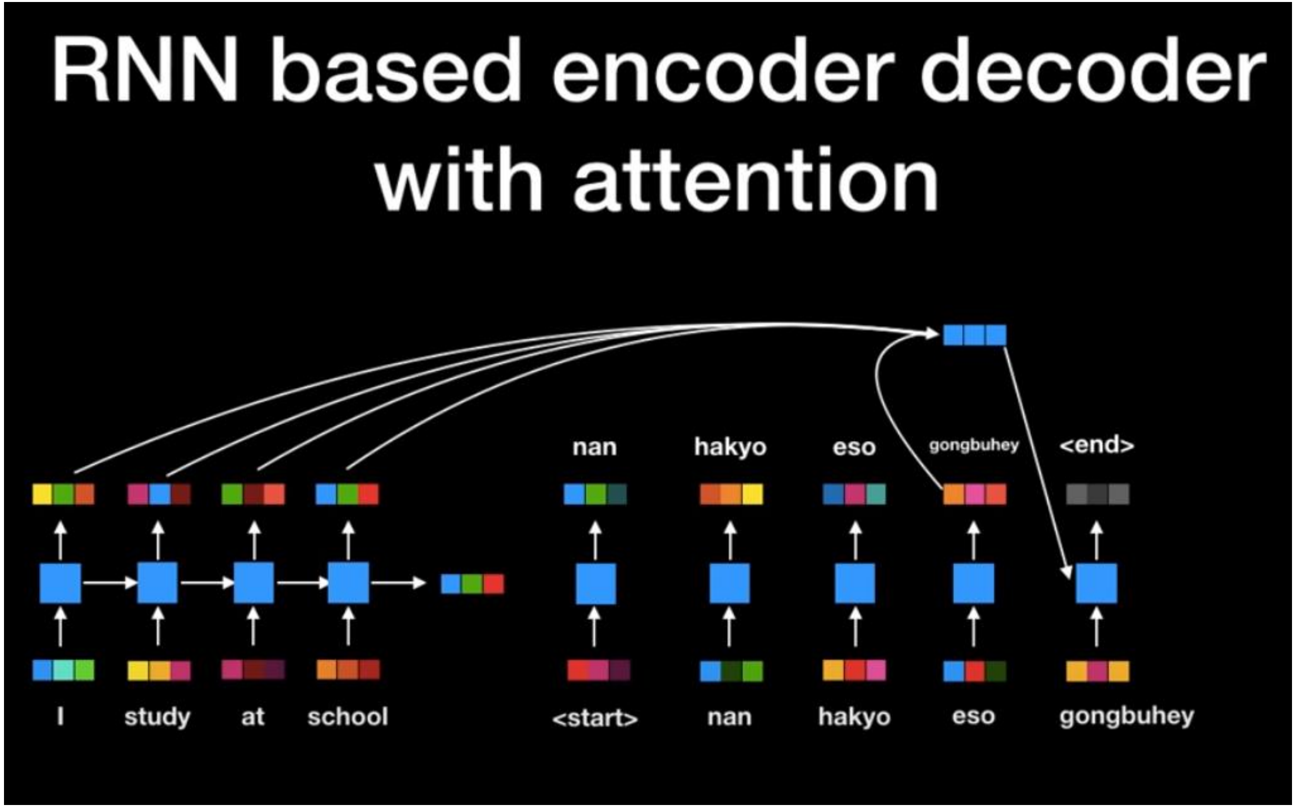

RNN based encoder decoder
with attention
0.01
0.13
0.01
0.85
nan
hakyo
eso
gongbuhey
<end>
I
study
at
school
<start>
nan
hakyo
eso
gongbuhey

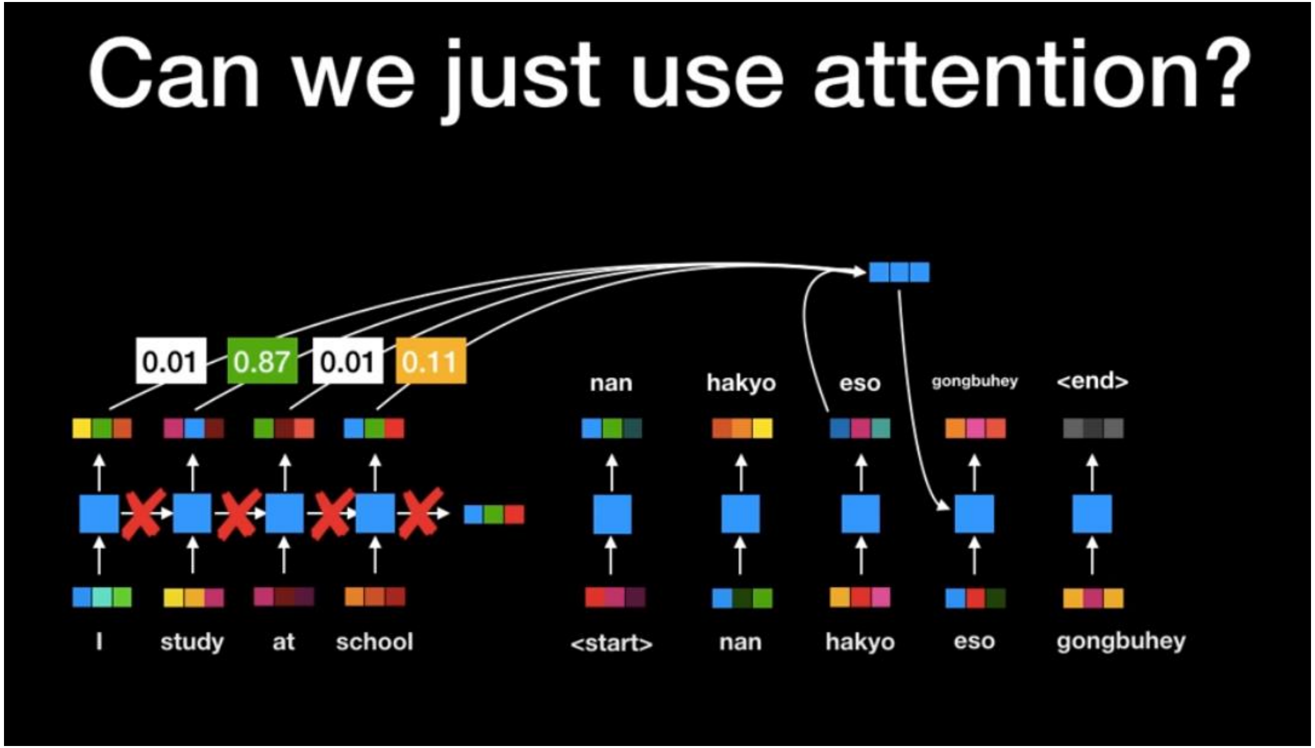
RNN based encoder decoder with attention



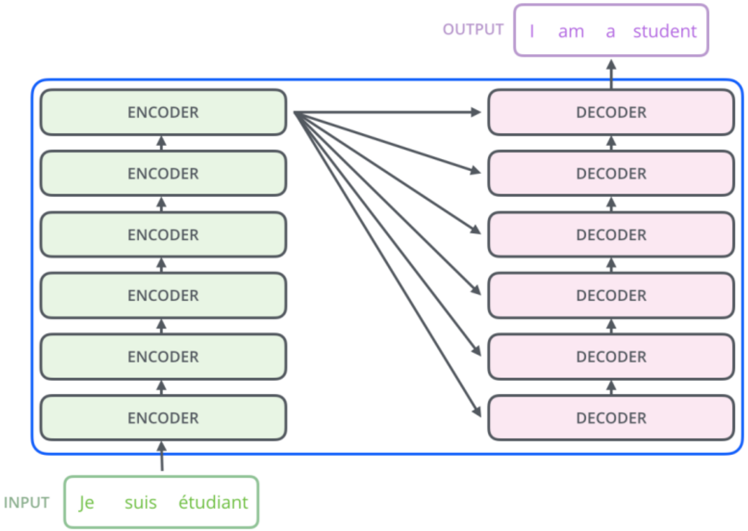
KU-BIG

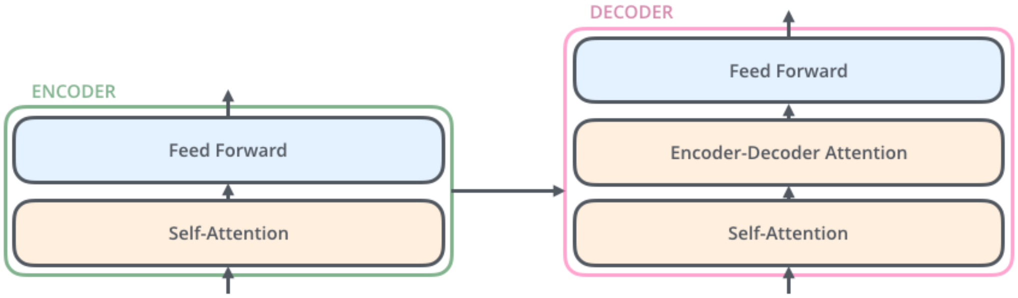

RNN based encoder decoder with attention
0.01
0.87
0.01
0.11
I
study
at
school
nan
hakyo
eso
gongbuhey
<end>
<start>
nan
hakyo
eso
gongbuhey

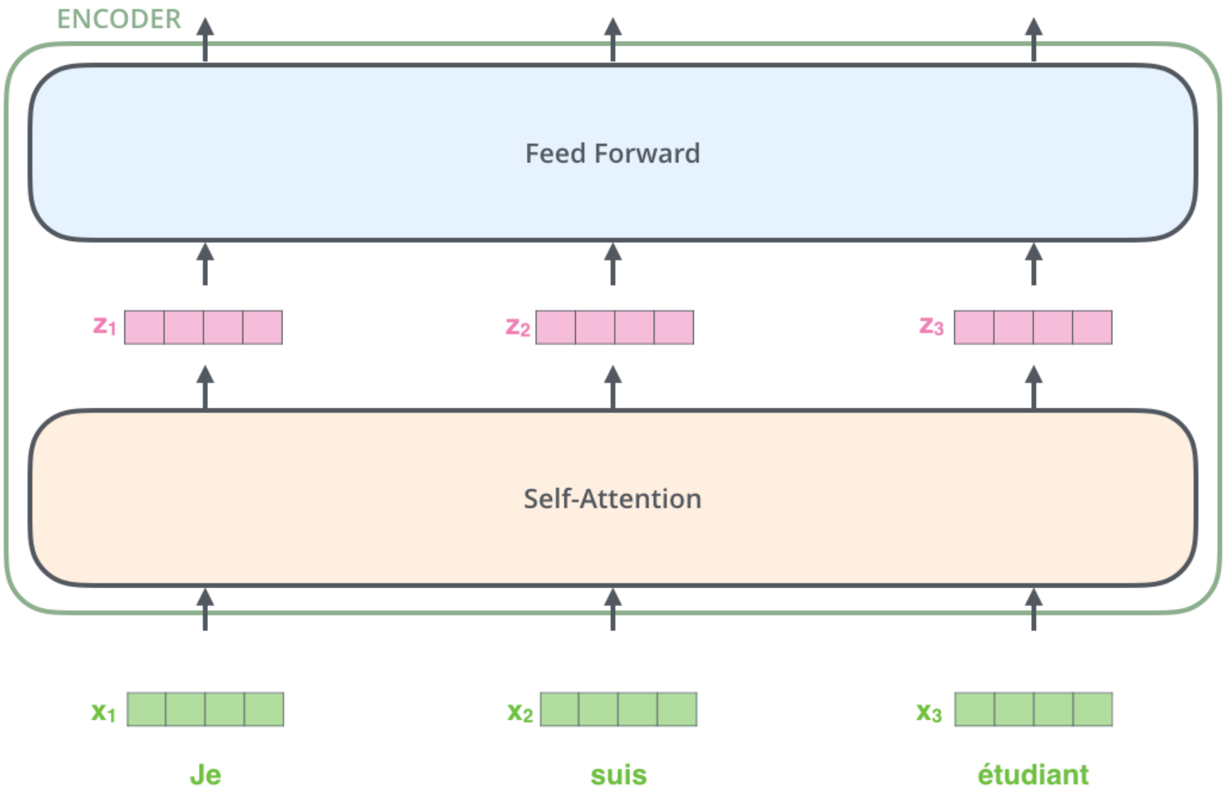

RNN based encoder decoder
with attention
nan
hakyo
eso
gongbuhey
<end>
I
study
at
school
<start>
nan
hakyo
eso
gongbuhey

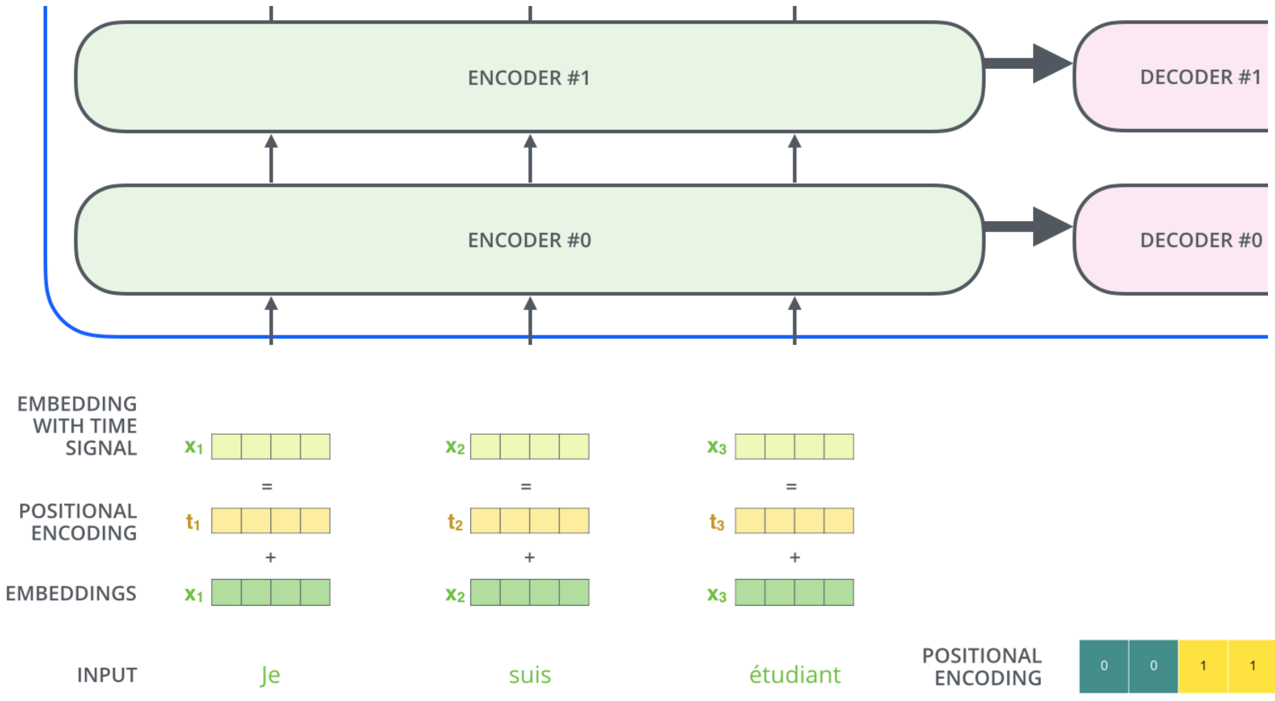

Can we just use attention?
0.01
0.87
0.01
0.11
nan
hakyo
eso
gongbuhey
<end>
I
study
at
school
<start>
nan
hakyo
eso
gongbuhey

Transformer

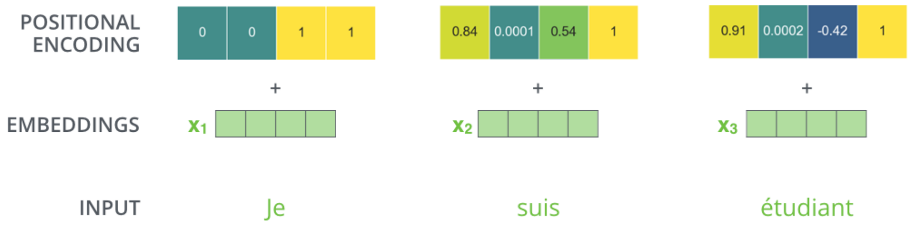





ENCODER
Feed Forward
z_1
z_2
z_3
Self-Attention
x_1
x_2
x_3
Je
suis
étudiant



ENCODER #1
DECODER #1
ENCODER #0
DECODER #0
EMBEDDING WITH TIME SIGNAL
POSITIONAL ENCODING
EMBEDDINGS
INPUT
Je
suis
étudiant



POSITIONAL ENCODING
0
0
1
1
0.84
0.0001
0.54
1
0.91
0.0002
-0.42
1
EMBEDDINGS
INPUT
Je
suis
étudiant

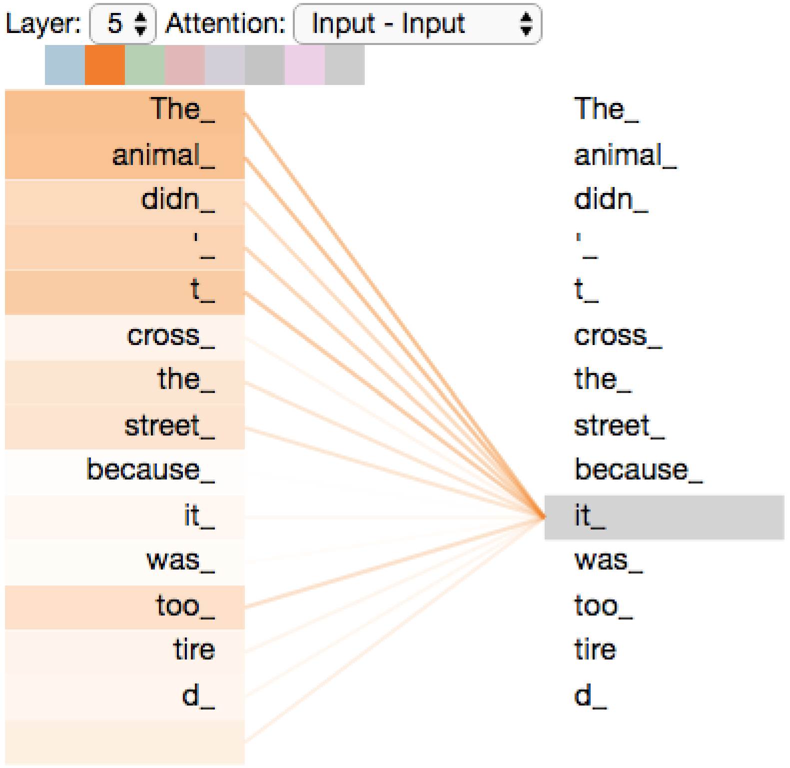
- Two properties that a good positional encoding scheme should have
  - ✓ The norm of encoding vector is the same for all positions
  - ✓ The further the two positions, the larger the distance
    - A Simple Example (n = 10, dim = 10)

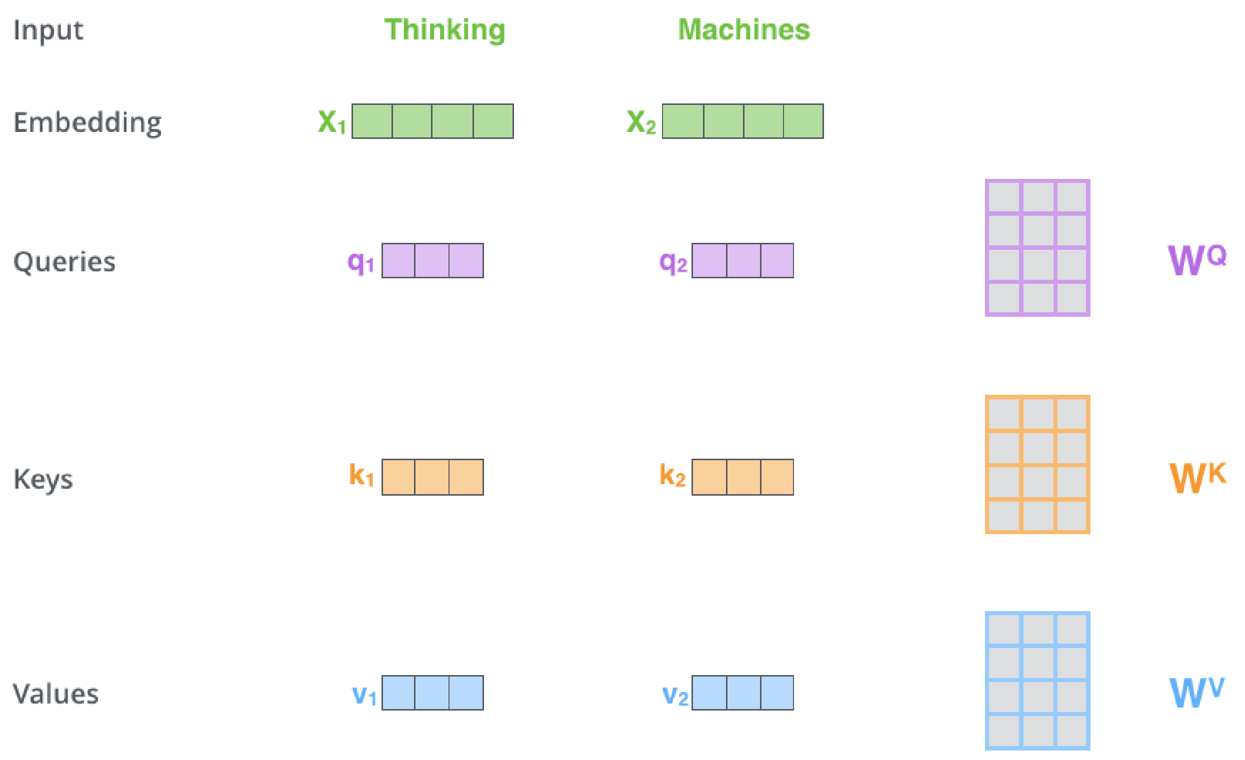
$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}})$$

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

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
X1	0.000	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000	1.000
X2	0.841	0.674	0.638	0.839	0.461	0.922	0.325	0.962	0.227	0.982
X3	0.909	-0.093	0.983	0.408	0.818	0.699	0.615	0.852	0.442	0.928
X4	0.141	-0.798	0.875	-0.155	0.991	0.368	0.838	0.678	0.634	0.841
X5	-0.757	-0.983	0.366	-0.668	0.942	-0.022	0.970	0.452	0.793	0.723
X6	-0.959	-0.526	-0.312	-0.965	0.680	-0.408	0.996	0.192	0.911	0.579
X7	-0.279	0.275	-0.847	-0.952	0.267	-0.730	0.915	-0.082	0.981	0.415
X8	0.657	0.896	-0.992	-0.632	-0.207	-0.938	0.734	-0.350	0.999	0.235
X9	0.989	0.932	-0.681	-0.109	-0.635	-0.999	0.473	-0.591	0.966	0.046
X10	0.412	0.360	-0.057	0.450	-0.919	-0.904	0.161	-0.788	0.882	-0.144

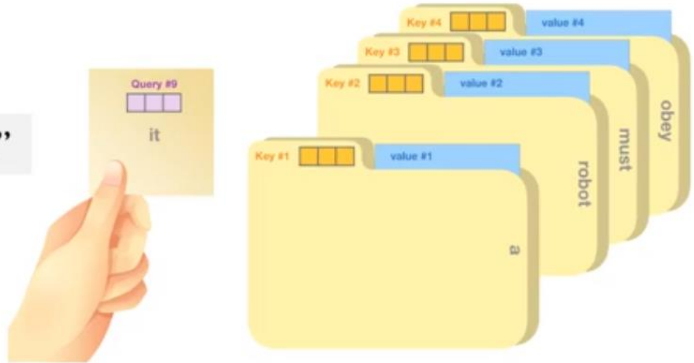
"The animal didn't cross the street because it was too tired"

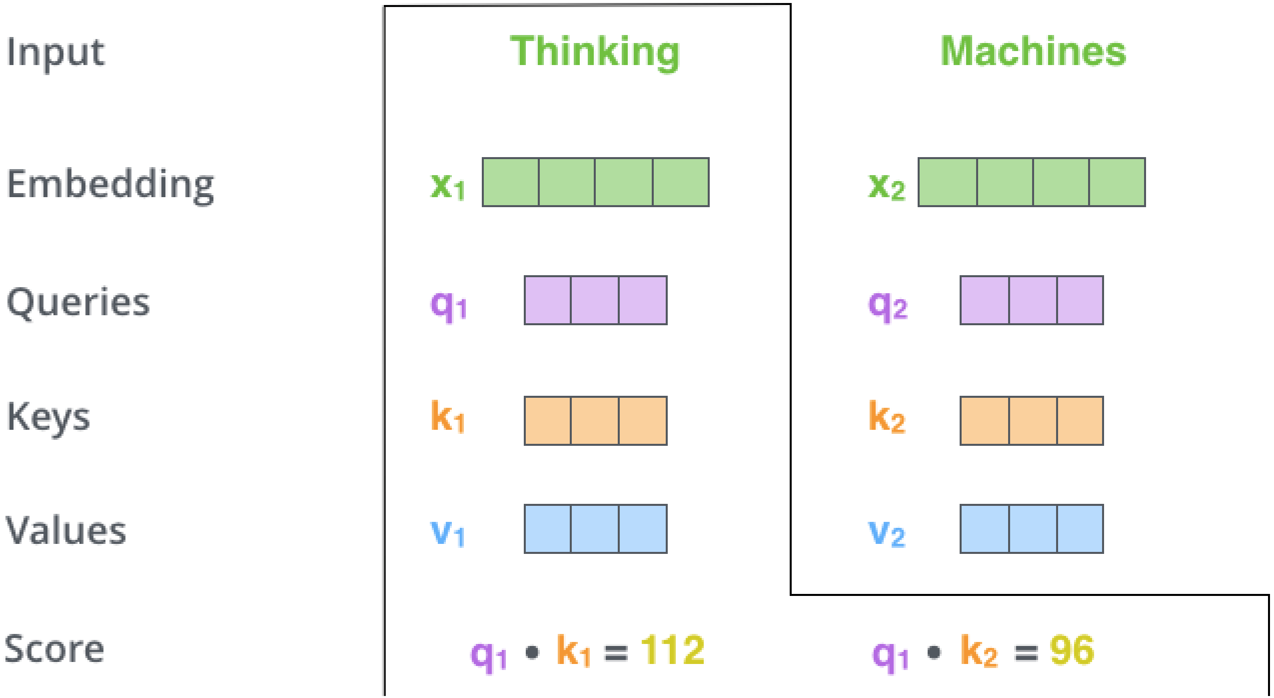



Input
Thinking
Machines
Embedding
X1
X2
Queries
q1
q2
WQ
Keys
k1
k2
WK
Values
v1
v2
WV

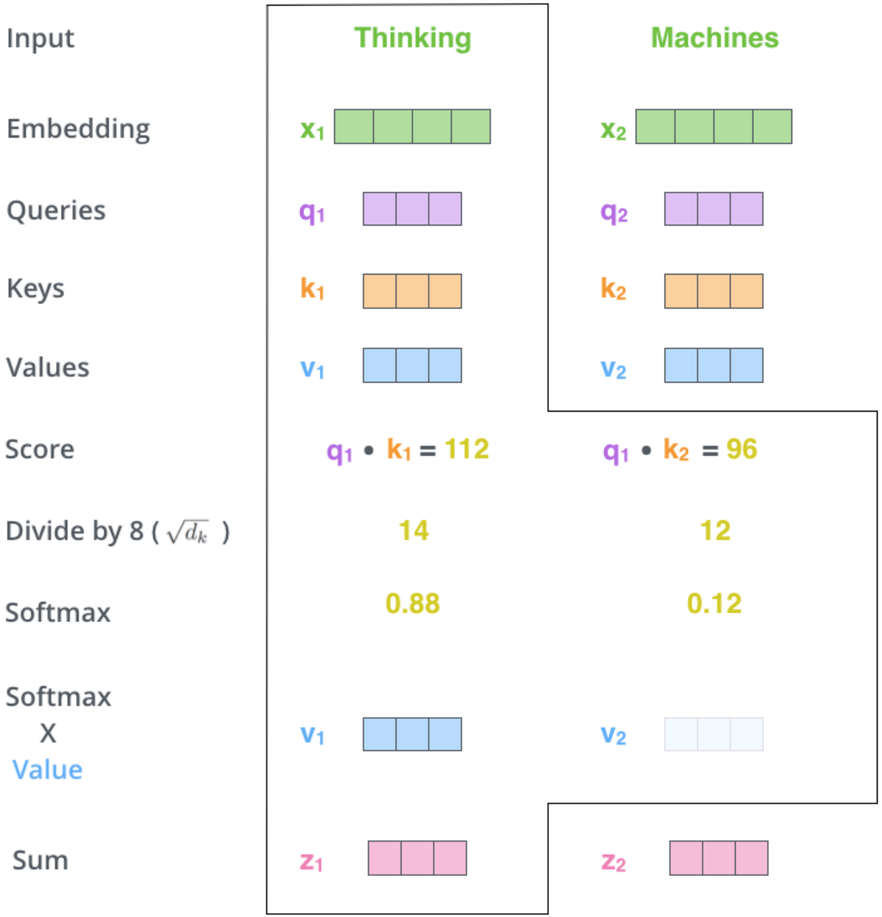
- Self-Attention in Detail
  - ✓ Step 1: Create three vectors from each of the encoder's input vectors
    - Query: The query is a representation of the current word used to score against all the other words (using their keys). We only care about the query of the token we're currently processing.
    - Key: Key vectors are like labels for all the words in the segment. They're what we match against in our search for relevant words.
    - Value: Value vectors are actual word representations, once we've scored how relevant each word is, these are the values we add up to represent the current word.

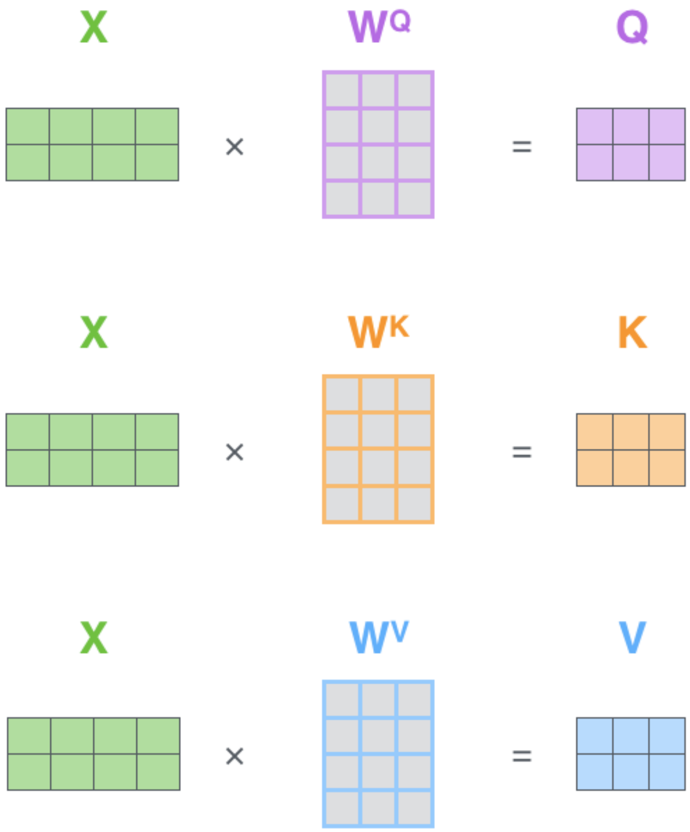
"A robot must obey the orders given it"

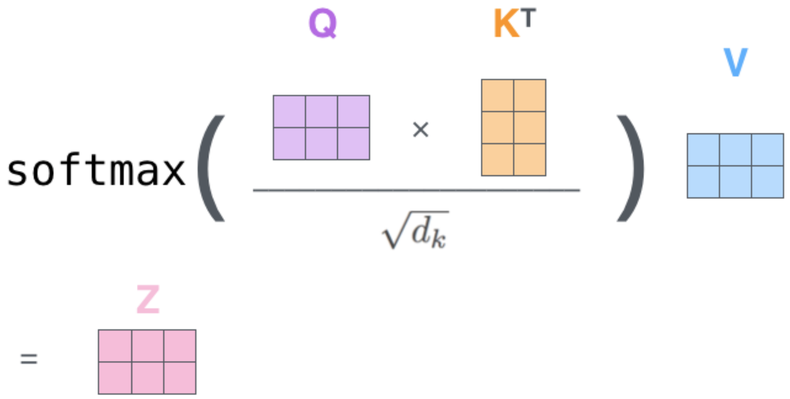


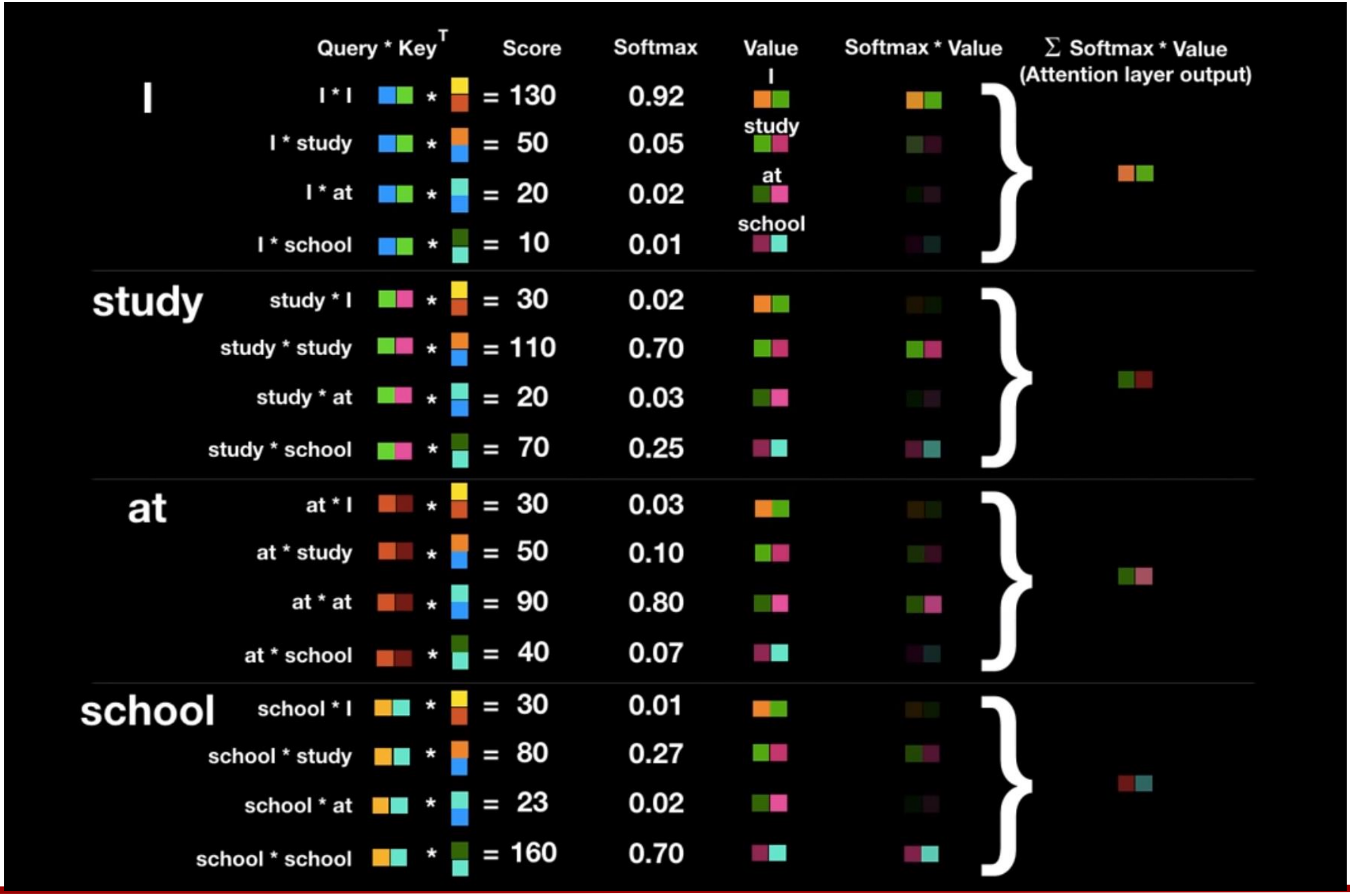

Input
Embedding
Queries
Keys
Values
Score
Thinking
Machines
x_1
x_2
q_1
q_2
k_1
k_2
v_1
v_2
$q_1 \cdot k_1 = 112$
$q_1 \cdot k_2 = 96$

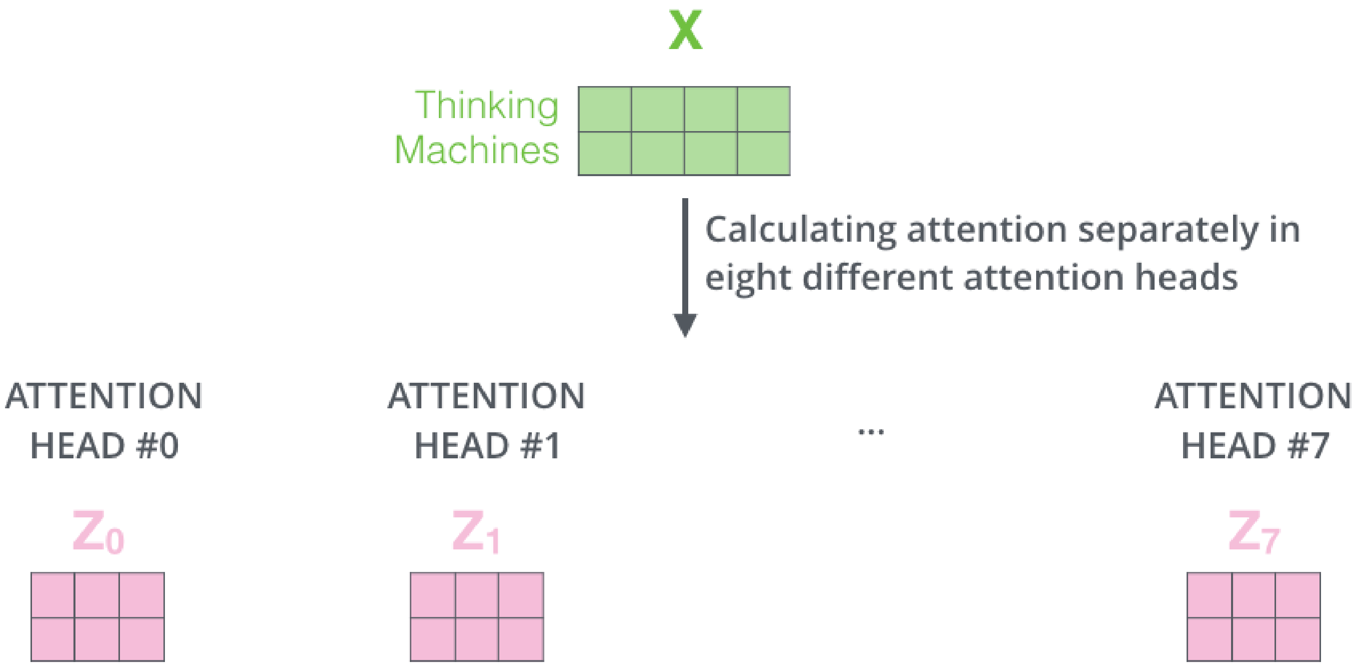
Input	Thinking	Machines
Embedding	x_1	x_2
Queries	q_1	q_2
Keys	k_1	k_2
Values	v_1	v_2
Score	$q_1 \cdot k_1 = 112$	$q_1 \cdot k_2 = 96$
Divide by 8 ($\sqrt{d_k}$)	14	12
Softmax	0.88	0.12


Input
Thinking
Machines
Embedding
x1
x2
Queries
q1
q2
Keys
k1
k2
Values
v1
v2
Score
q1 • k1 = 112
q1 • k2 = 96
Divide by 8 ($\sqrt{d_k}$)
14
12
Softmax
0.88
0.12
Softmax
X
Value
v1
v2
Sum
z1
z2

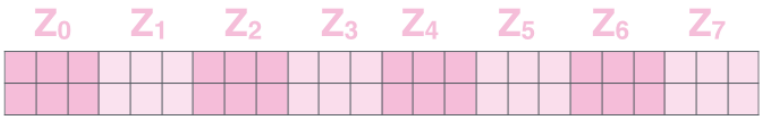

X
W^Q
Q
X
W^K
K
X
W^V
V


softmax
Q
K^T
V
$\sqrt{d_k}$
Z


Query * Key T
Score
Softmax
Value
Softmax * Value
Σ Softmax * Value
(Attention layer output)
I
I * I = 130 0.92
I * study = 50 0.05
I * at = 20 0.02
I * school = 10 0.01
I
study
at
school
study
study * I = 30 0.02
study * study = 110 0.70
study * at = 20 0.03
study * school = 70 0.25
at
at * I = 30 0.03
at * study = 50 0.10
at * at = 90 0.80
at * school = 40 0.07
school
school * I = 30 0.01
school * study = 80 0.27
school * at = 23 0.02
school * school = 160 0.70

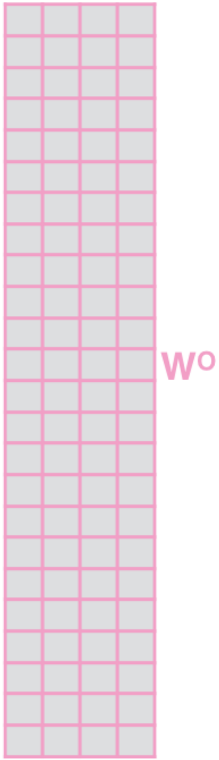

X
Thinking
Machines
Calculating attention separately in eight different attention heads
ATTENTION HEAD #0
ATTENTION HEAD #1
...
ATTENTION HEAD #7
Z_0
Z_1
Z_7

1) Concatenate all the attention heads



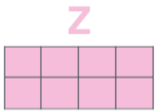
2) Multiply with a weight matrix W^O that was trained jointly with the model

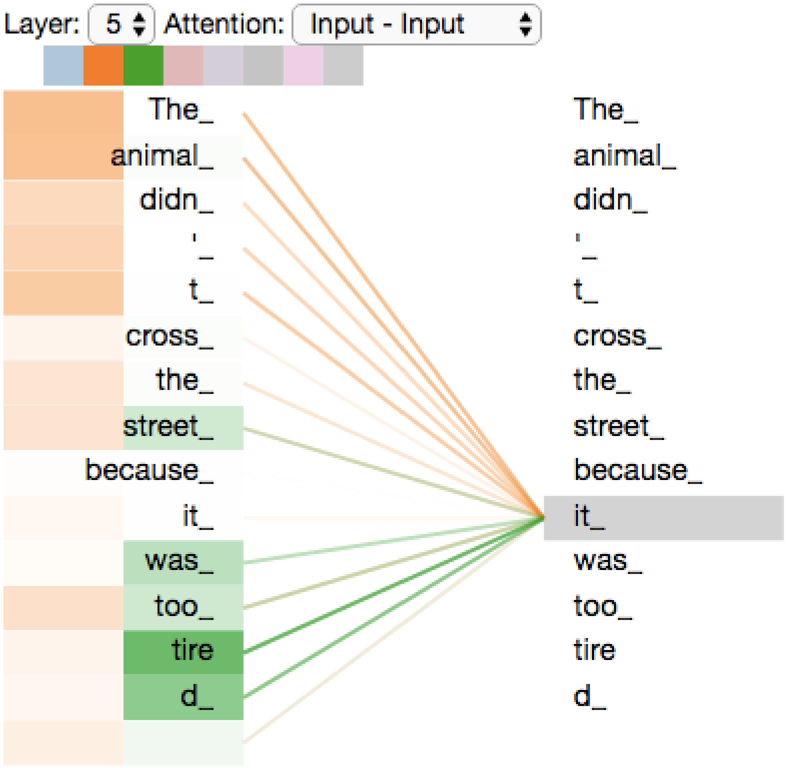
X

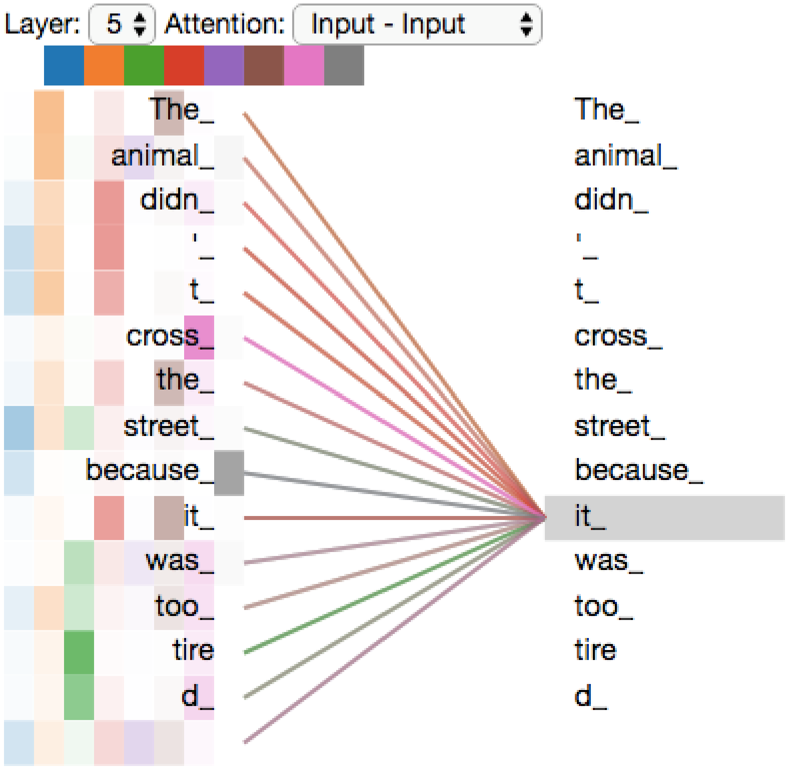


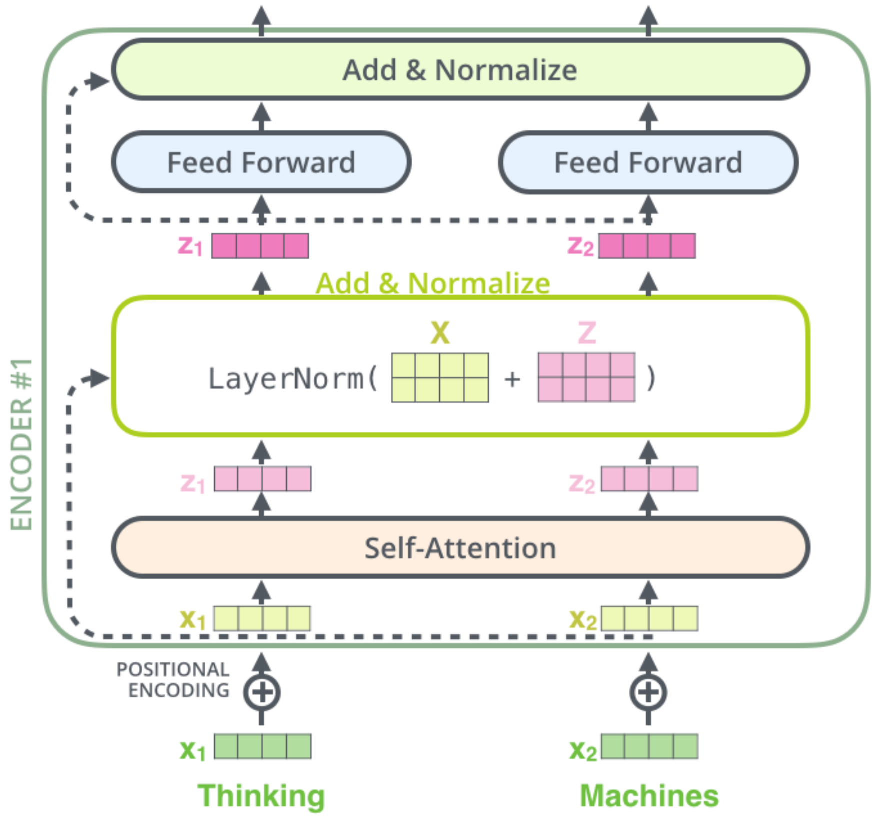
3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN

=

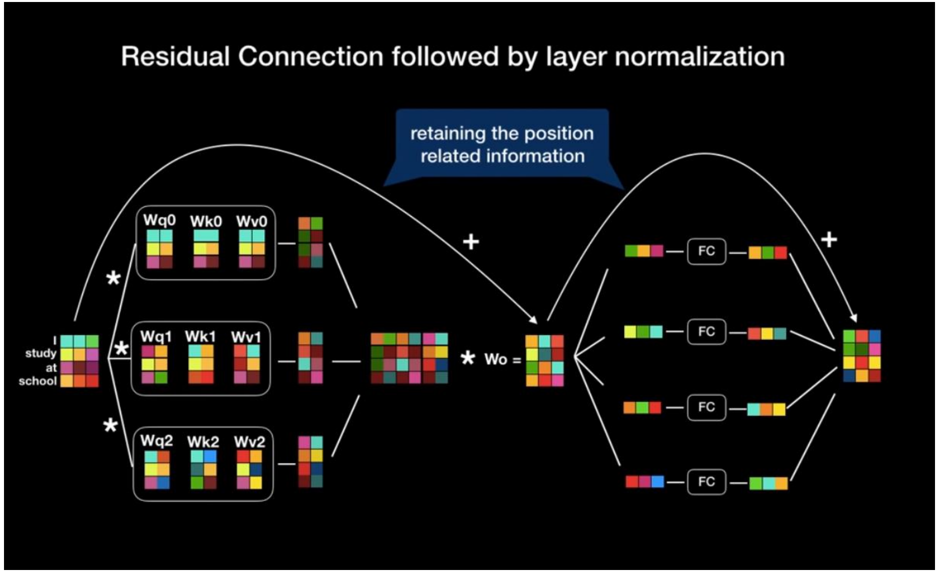



Layer: 5
Attention: Input - Input
The_
animal_
didn_
'_
t_
cross_
the_
street_
because_
it_
was_
too_
tire
d_


Layer: 5
Attention: Input - Input
The_
animal_
didn_
'_
t_
cross_
the_
street_
because_
it_
was_
too_
tire
d_



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

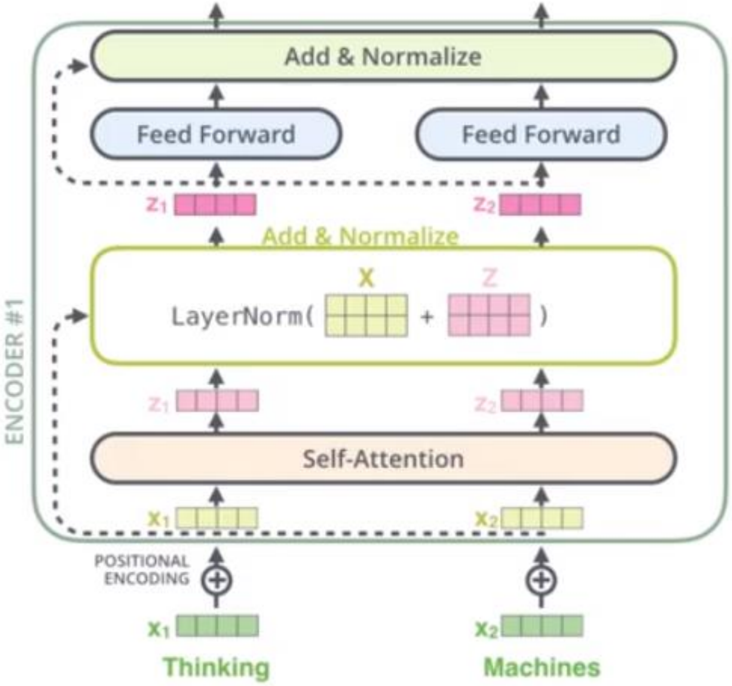


Residual Connection followed by layer normalization
retaining the position
related information
Wq0 Wk0 Wv0
Wq1 Wk1 Wv1
Wq2 Wk2 Wv2
I
study
at
school
* Wo =
FC
FC
FC
FC

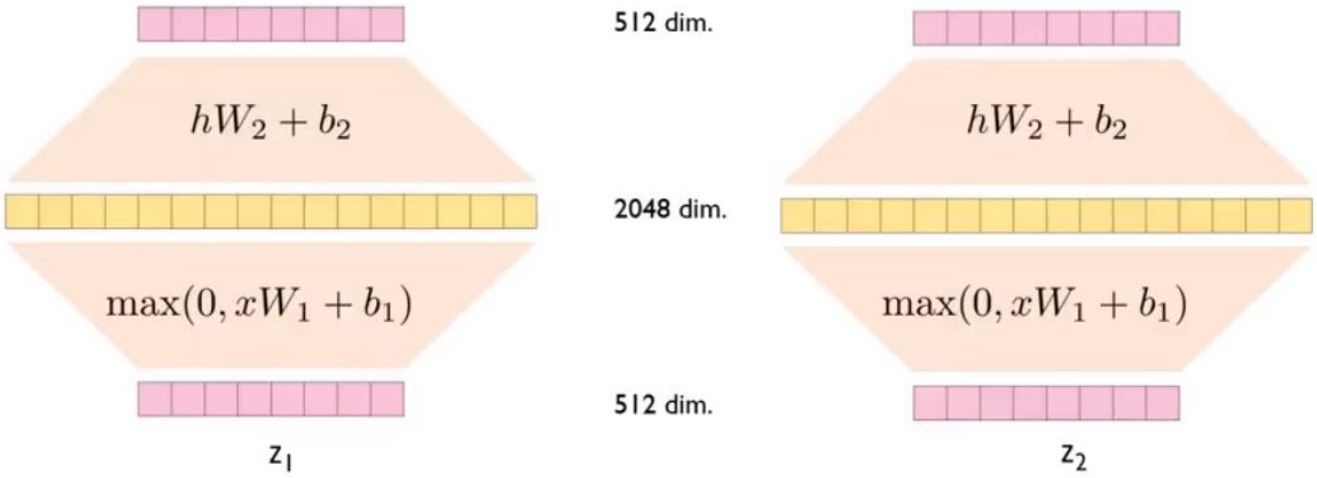
- Position-wise Feed-Forward Networks
  - ✓ Fully connected feed-forward network
  - ✓ Applied to each position separately and identically

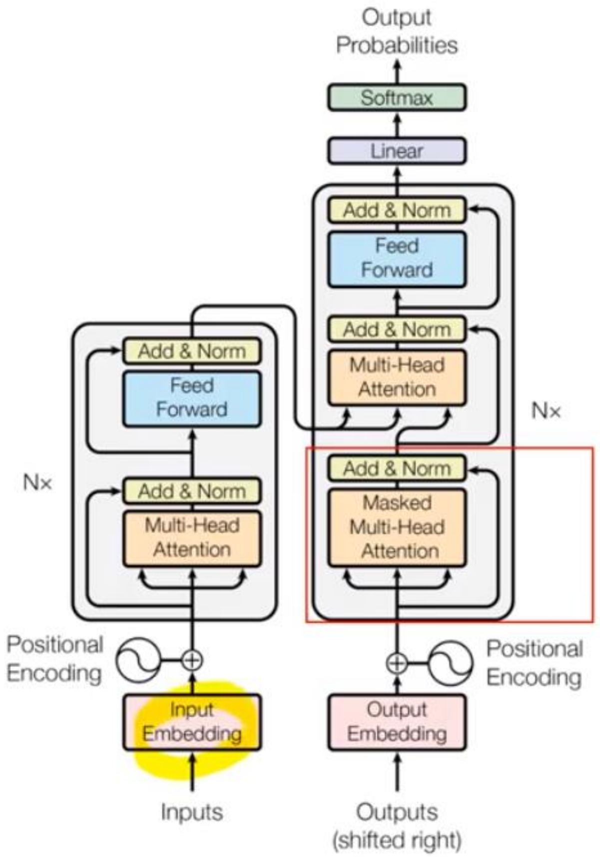
$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

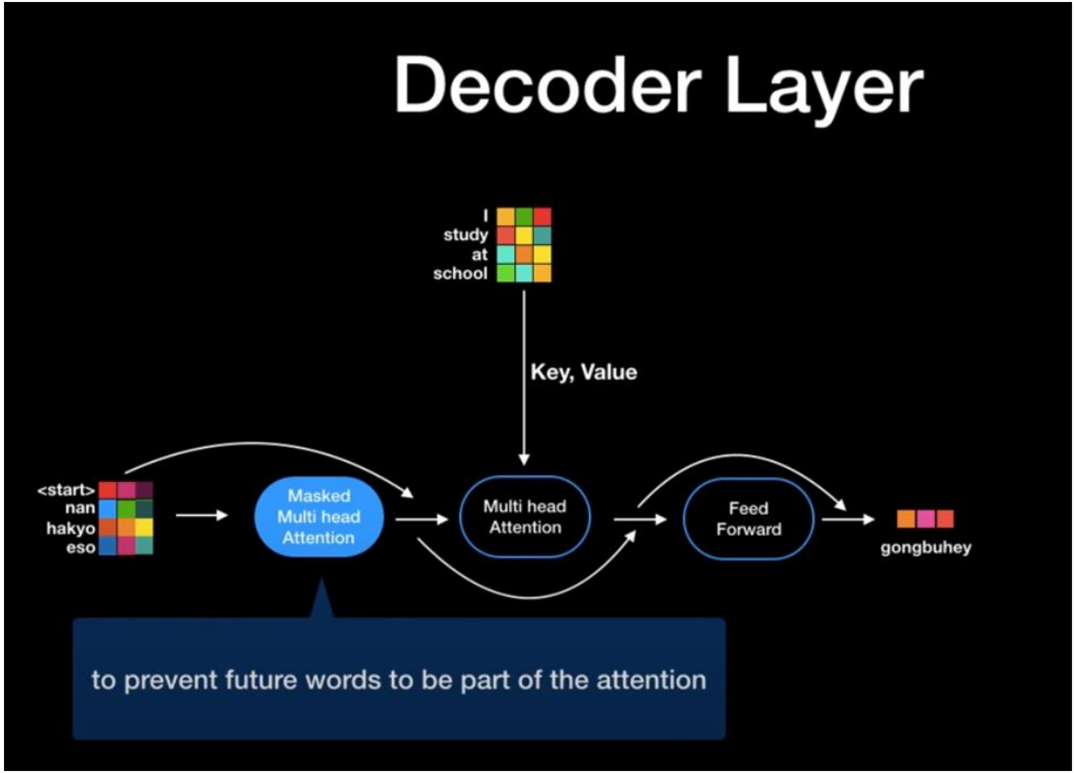
  - ✓ The linear transformations are the same across different positions
  - ✓ They use different parameters from layer to layer



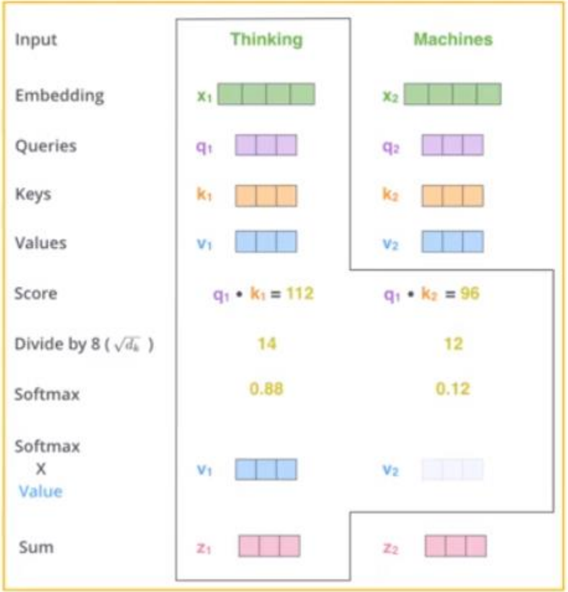
- Position-wise Feed-Forward Networks

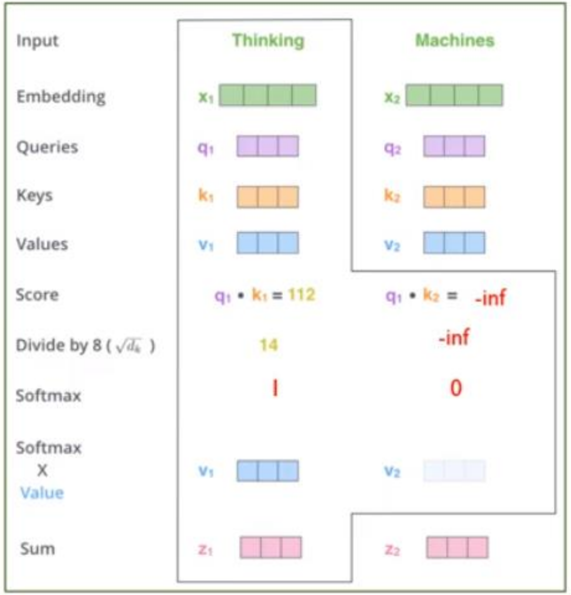



Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Add & Norm
Masked
Multi-Head
Attention
Nx
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Positional
Encoding
Input
Embedding
Output
Embedding
Inputs
Outputs
(shifted right)


Decoder Layer
I
study
at
school
Key, Value
<start>
nan
hakyo
eso
Masked
Multi head
Attention
Multi head
Attention
Feed
Forward
gongbuhey
to prevent future words to be part of the attention

- Masked Multi-head Attention
  - ✓ Self attention layers in the decoder is only allowed to attend to earlier positions in the output sequence, which is done by masking future positions (setting them to –inf) before the softmax step in the self attention calculation.





- Masked Multi-head Attention

Queries

robot	must	obey	orders

X

Keys

robot	must	obey	orders
robot	must	obey	orders
robot	must	obey	orders
robot	must	obey	orders

=

Scores
(before softmax)

0.11	0.00	0.81	0.79
0.19	0.50	0.30	0.48
0.53	0.98	0.95	0.14
0.81	0.86	0.38	0.90

Scores
(before softmax)

0.11	0.00	0.81	0.79
0.19	0.50	0.30	0.48
0.53	0.98	0.95	0.14
0.81	0.86	0.38	0.90

Apply Attention Mask →

Masked Scores
(before softmax)

0.11	-inf	-inf	-inf
0.19	0.50	-inf	-inf
0.53	0.98	0.95	-inf
0.81	0.86	0.38	0.90

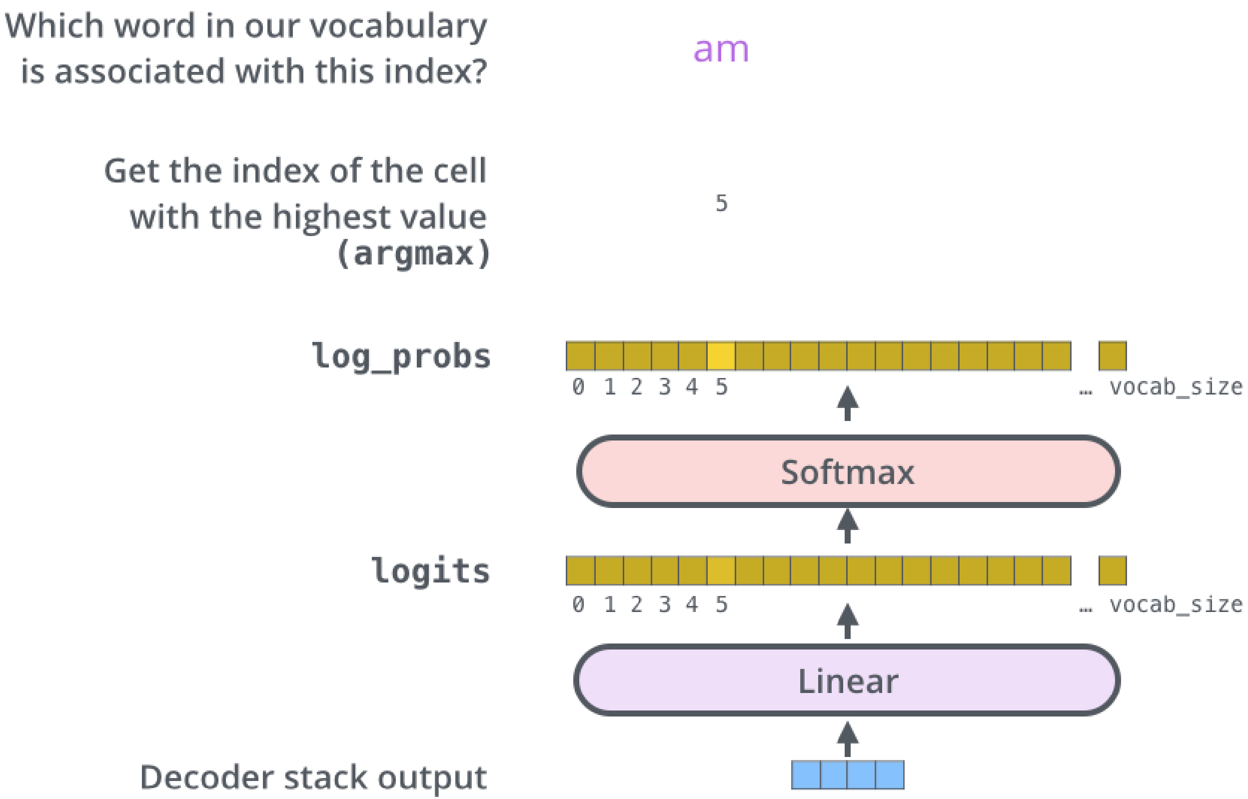
Masked Scores
(before softmax)

0.11	-inf	-inf	-inf
0.19	0.50	-inf	-inf
0.53	0.98	0.95	-inf
0.81	0.86	0.38	0.90

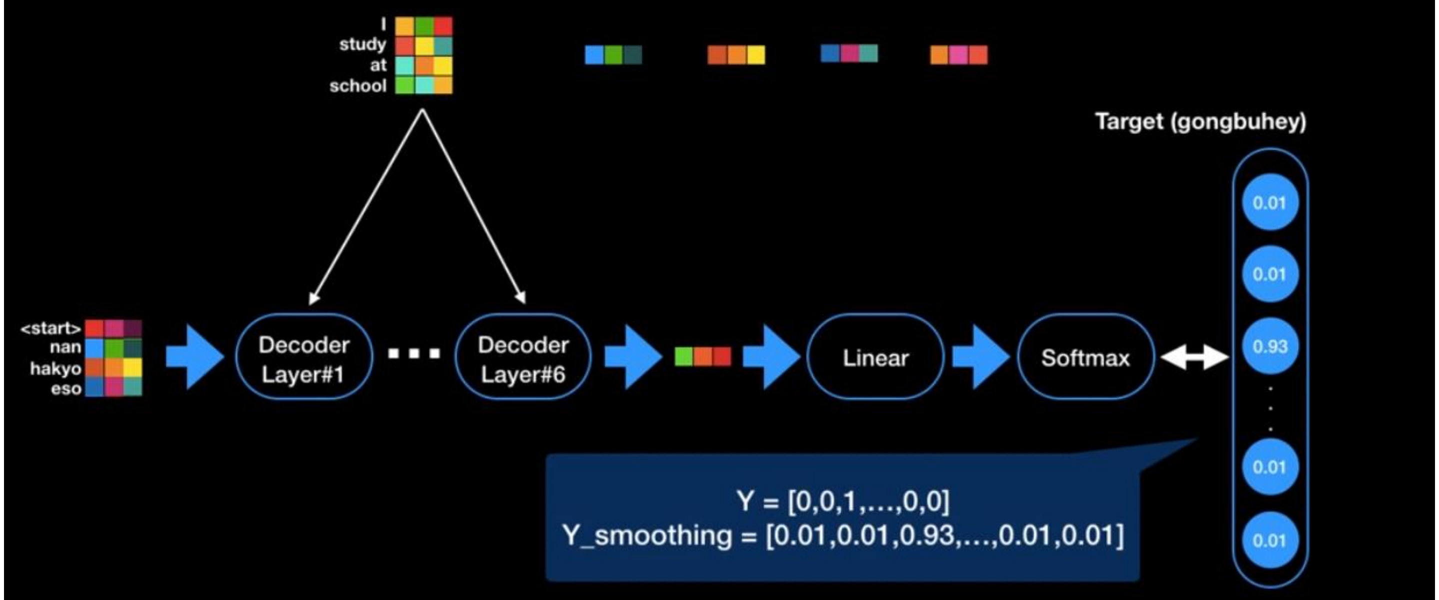
Softmax (along rows) →

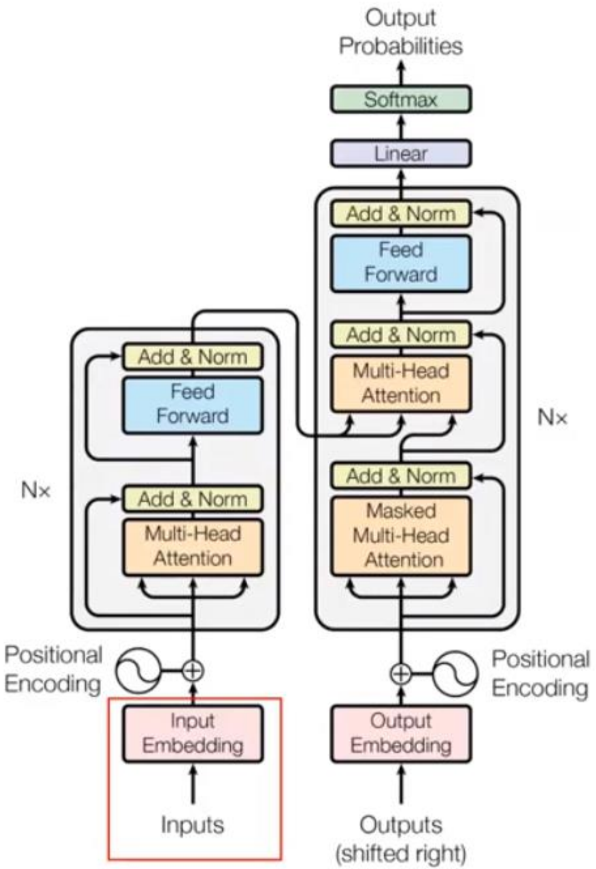
Scores

1	0	0	0
0.48	0.52	0	0
0.31	0.35	0.34	0
0.25	0.26	0.23	0.26


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

Label Smoothing




Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Add & Norm
Masked
Multi-Head
Attention
Nx
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Positional
Encoding
Input
Embedding
Output
Embedding
Inputs
Outputs
(shifted right)

- Performance (in terms of BLEU score)

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

- Transformer variations

	N	d_{model}	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_{ls}	train steps	PPL (dev)	BLEU (dev)	params $\times 10^6$
base	6	512	2048	8	64	64	0.1	0.1	100K	4.92	25.8	65
(A)				1	512	512				5.29	24.9	
				4	128	128				5.00	25.5	
				16	32	32				4.91	25.8	
				32	16	16				5.01	25.4	
(B)					16					5.16	25.1	58
					32					5.01	25.4	60
(C)	2									6.11	23.7	36
	4									5.19	25.3	50
	8									4.88	25.5	80
		256			32	32				5.75	24.5	28
		1024			128	128				4.66	26.0	168
			1024							5.12	25.4	53
			4096							4.75	26.2	90
(D)							0.0			5.77	24.6	
							0.2			4.95	25.5	
								0.0		4.67	25.3	
								0.2		5.47	25.7	
(E)	positional embedding instead of sinusoids									4.92	25.7	
big	6	1024	4096	16			0.3		300K	**4.33**	**26.4**	213

January 16, 2013

WORD2VEC

Word2Vec Paper by Tomas Mikolov et al

January 2, 2014

GLOVE

GloVe: Global Vectors for Word Representation by Jeffrey Pennington et al.

July 15, 2016

FASTTEXT

Enriching Word Vectors with Subword Information by Piotr Bojanowski et al

June 12, 2017

TRANSFORMER

Attention Is All You Need by Ashish Vaswani et al

February 15, 2018

ELMO

Deep contextualized word representations by Matthew E. Peters et al

October 11, 2018

BERT

BERT: Pre-training of Deep Bidirectional Transformers for...

reference

- http://jalammar.github.io/illustrated-transformer/
- Attention Is All You Need https://arxiv.org/abs/1706.03762