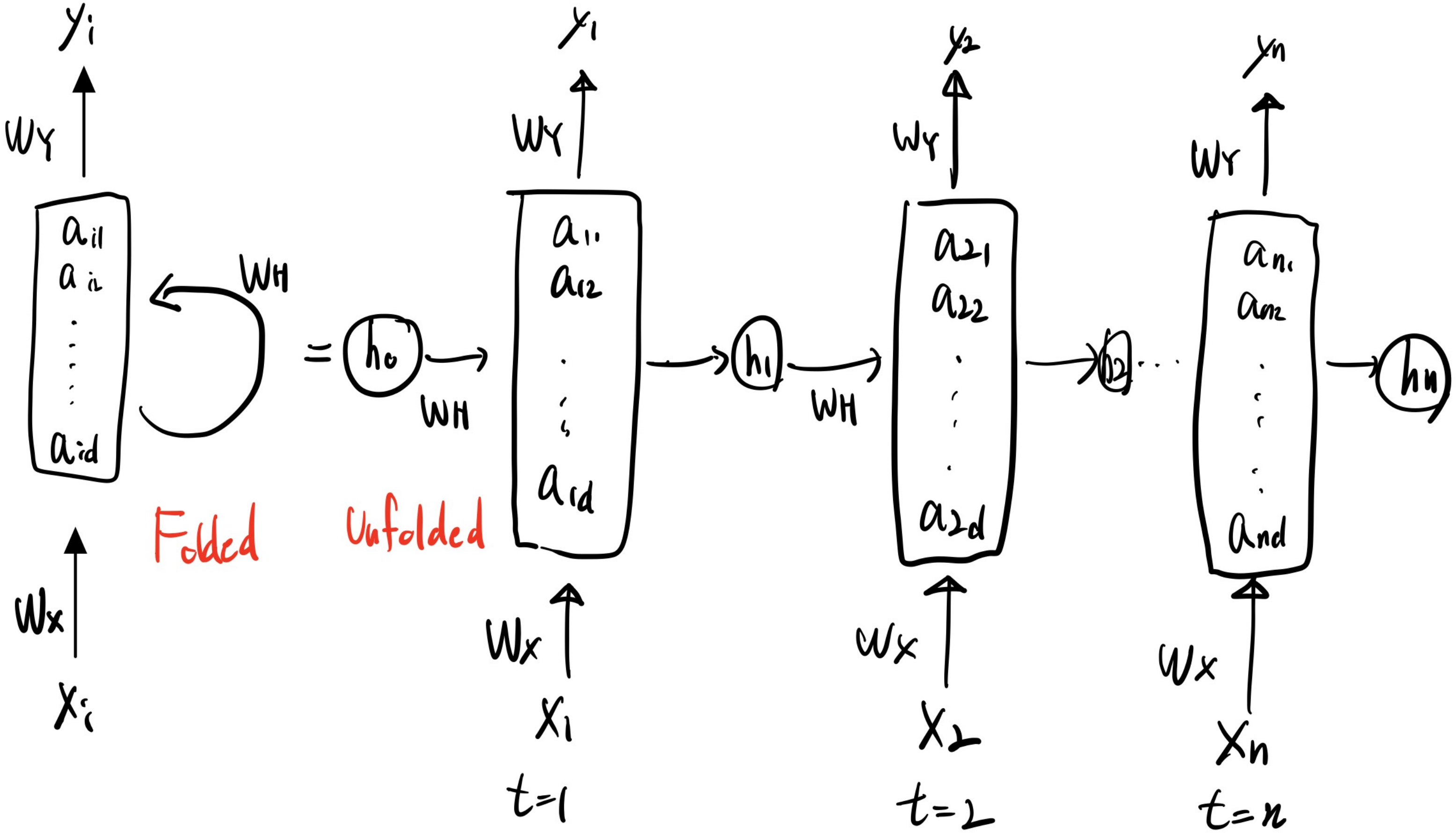
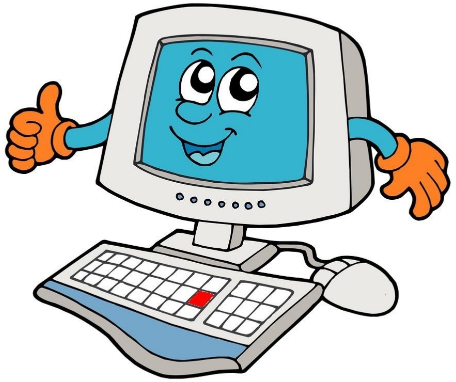
Recurrent Neural Network

: Processing Sequence Data

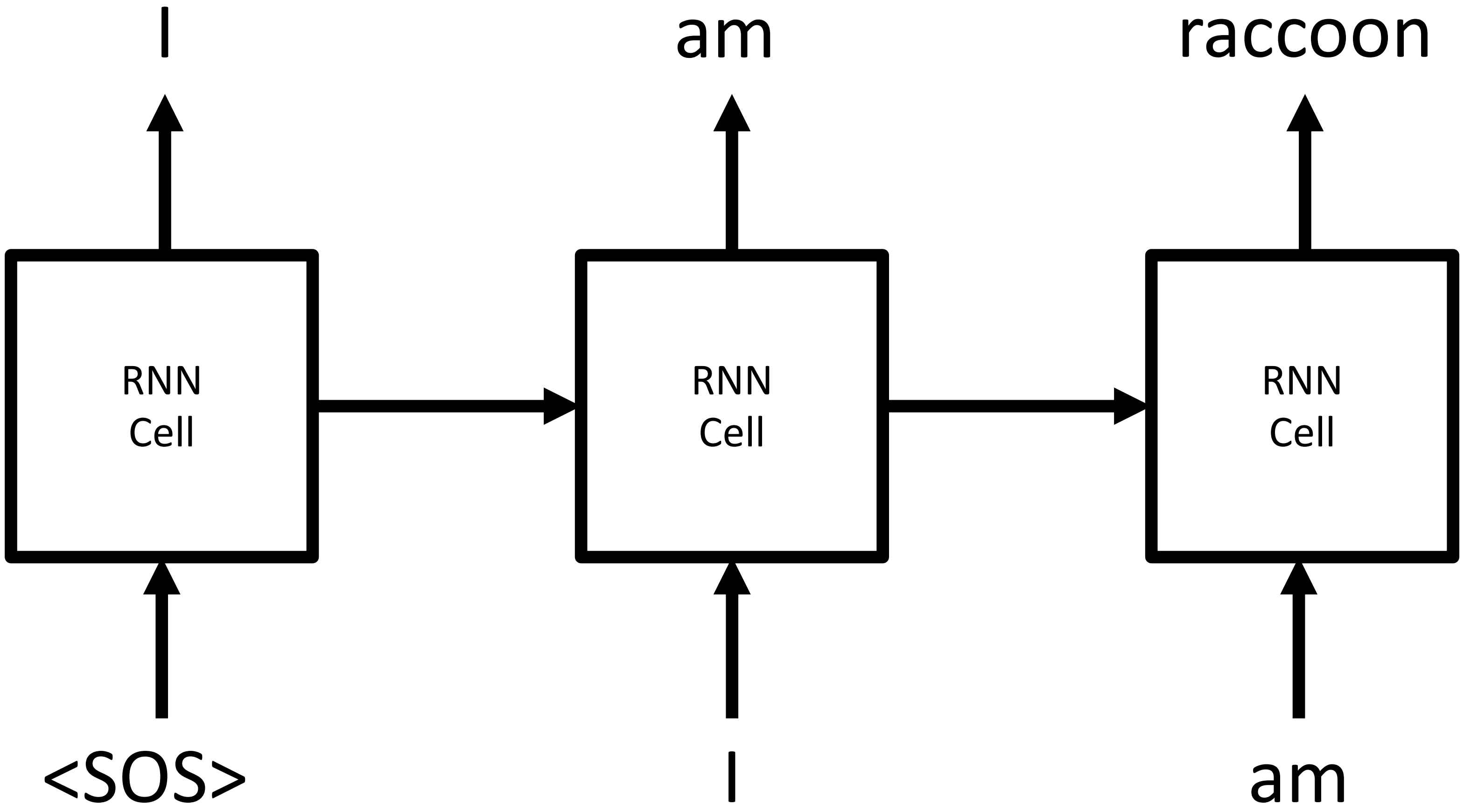
Reminder

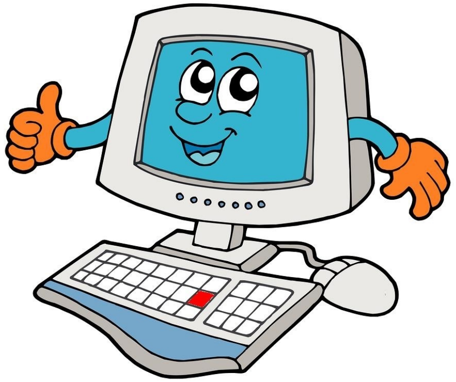


yi
WY
ai1
ai2
aid
WH
Folded
Wx
Xi
= h0
WH
Unfolded
y1
WY
a11
a12
a1d
Wx
X1
t=1
h1
WH
y2
WY
a21
a22
a2d
Wx
X2
t=2
h2
yn
WY
an1
an2
and
Wx
Xn
t=n
hn

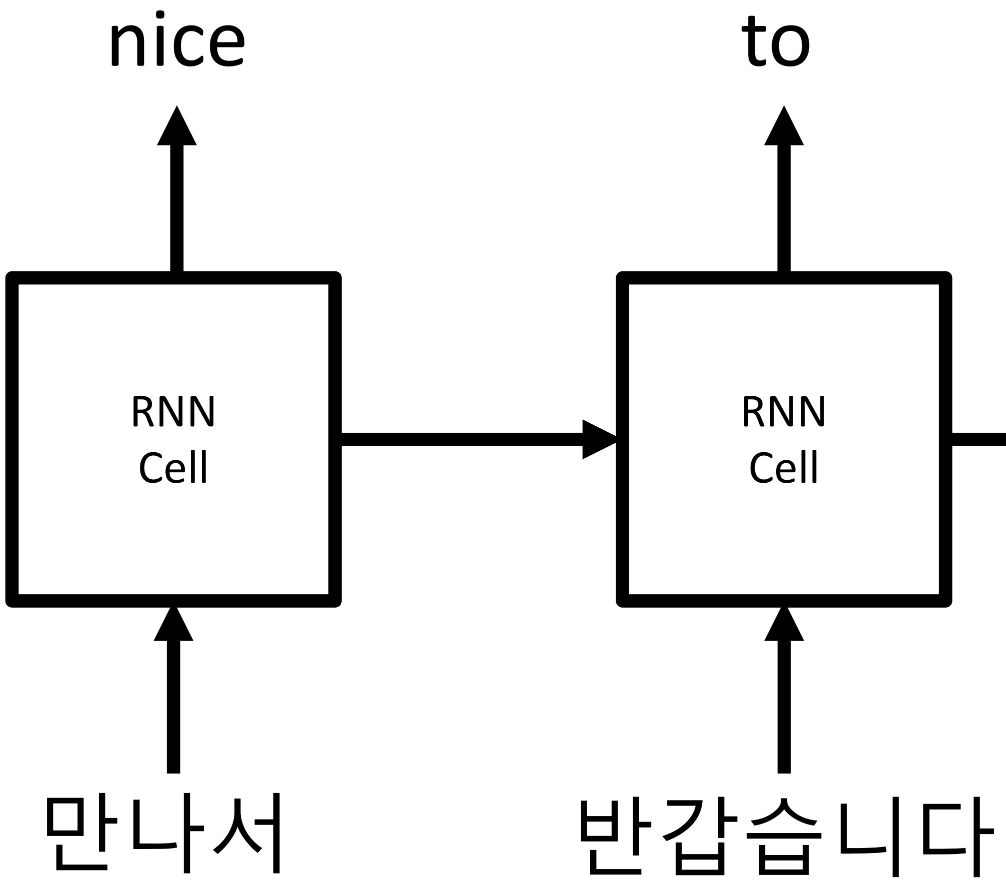


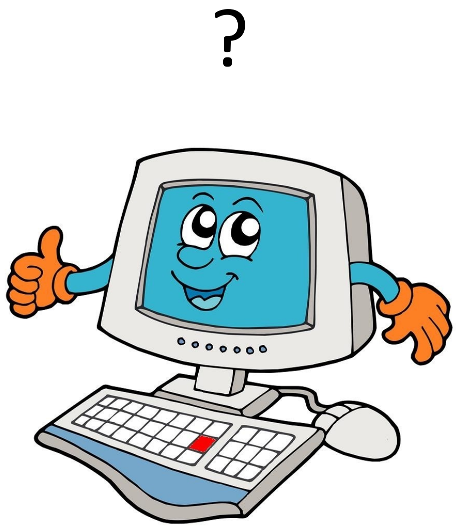
Text Generation





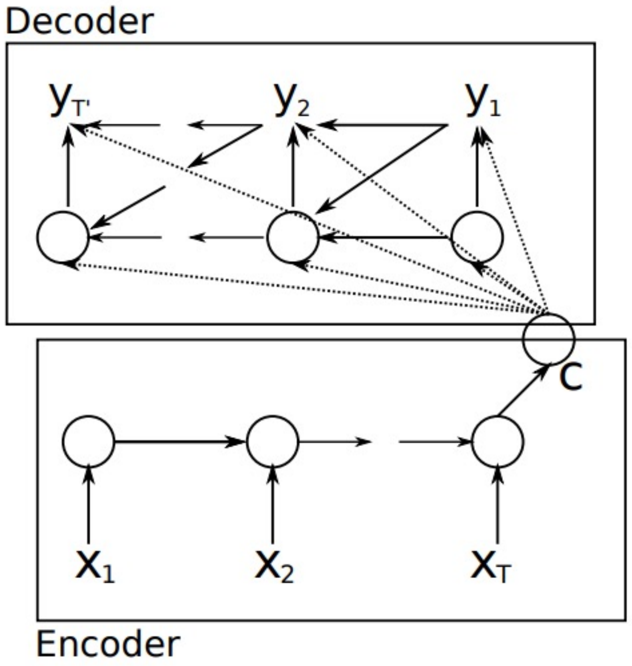
Machine Translation Example





RNN Encoder-Decoder Model:

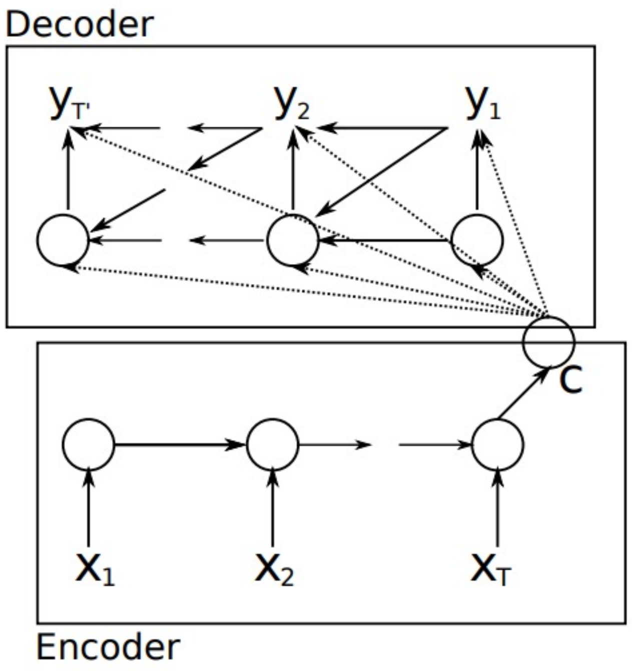
Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation (2014, Cho et al.)



Main objective of Encoder-decoder model:

Text Generation

RNN Encoder-Decoder Model:
Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation (2014, Cho et al.)



$p(y_1, \ldots, y_{T'} | x_1, \ldots, x_T)$

given input sequence x, **model computes *output sequence y's conditional distribution.***
T != T'

$h_{<t>} = f(h_{<t-1>}, y_{t-1}, c)$

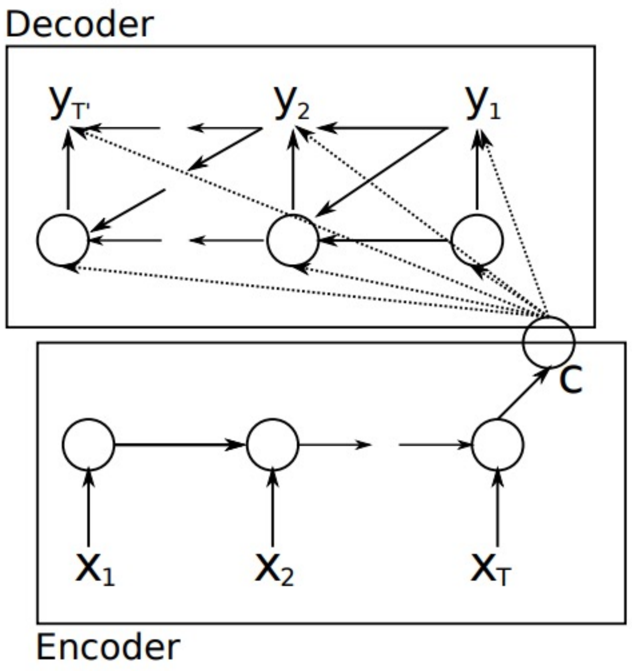
h_<t> = decoder hidden state at time t
c = encoders' summary vector

$P(y_t | y_{t-1}, y_{t-2}, \ldots, y_1, c) = g(h_{<t>}, y_{t-1}, c)$

f and g is activation function.
g must produce probabilities.

RNN Encoder-Decoder Model:

Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation (2014, Cho et al.)

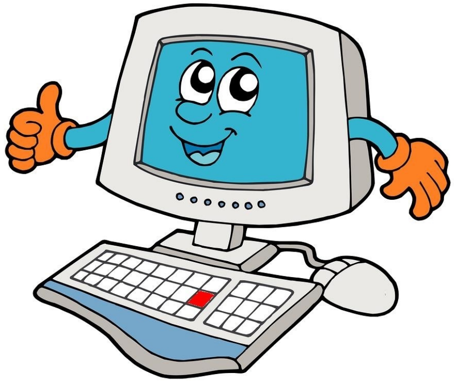


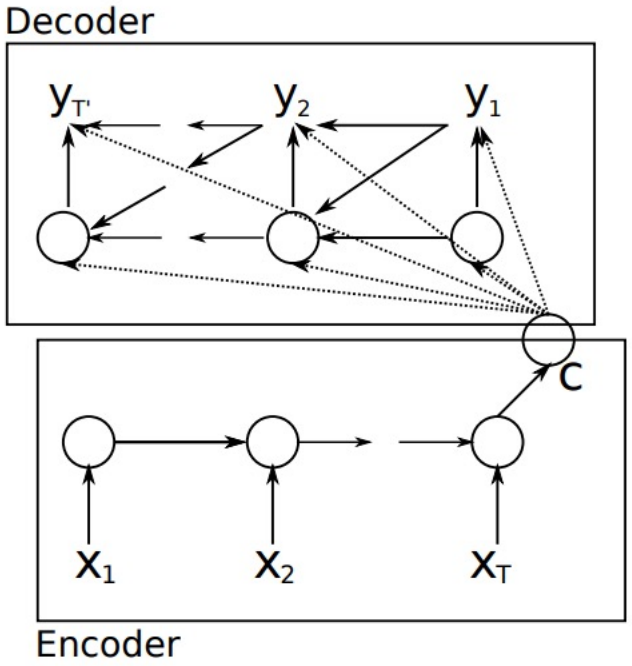
$$max_{\theta} \frac{1}{N} \sum_{n=1}^{N} log\, p_{\theta}(y_n|x_n)$$

two components of the proposed model are jointly trained to **maximize the conditional log-likelihood**

RNN Encoder-Decoder

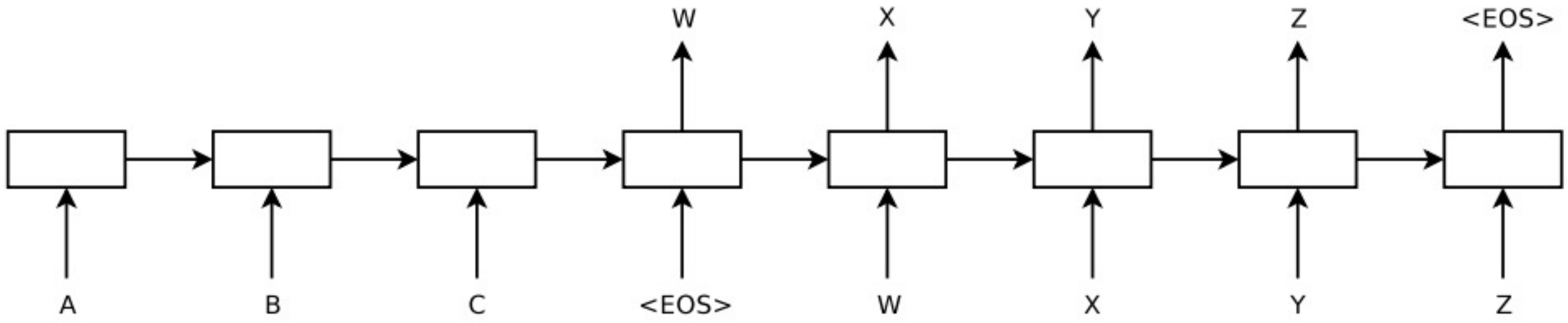
nice to meet you





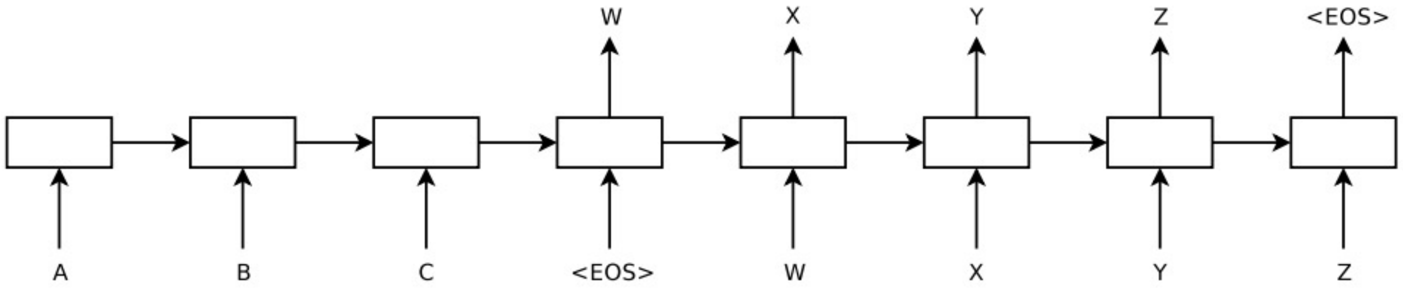
만나서 반갑습니다

Sequence to Sequence Learning with Neural Networks (2014, Sutskever et al.)



input sequence: ABC -> encoder
(objective) output sequence: WXYZ -> decoder

Sequence to Sequence Learning with Neural Networks (2014, Sutskever et al.)

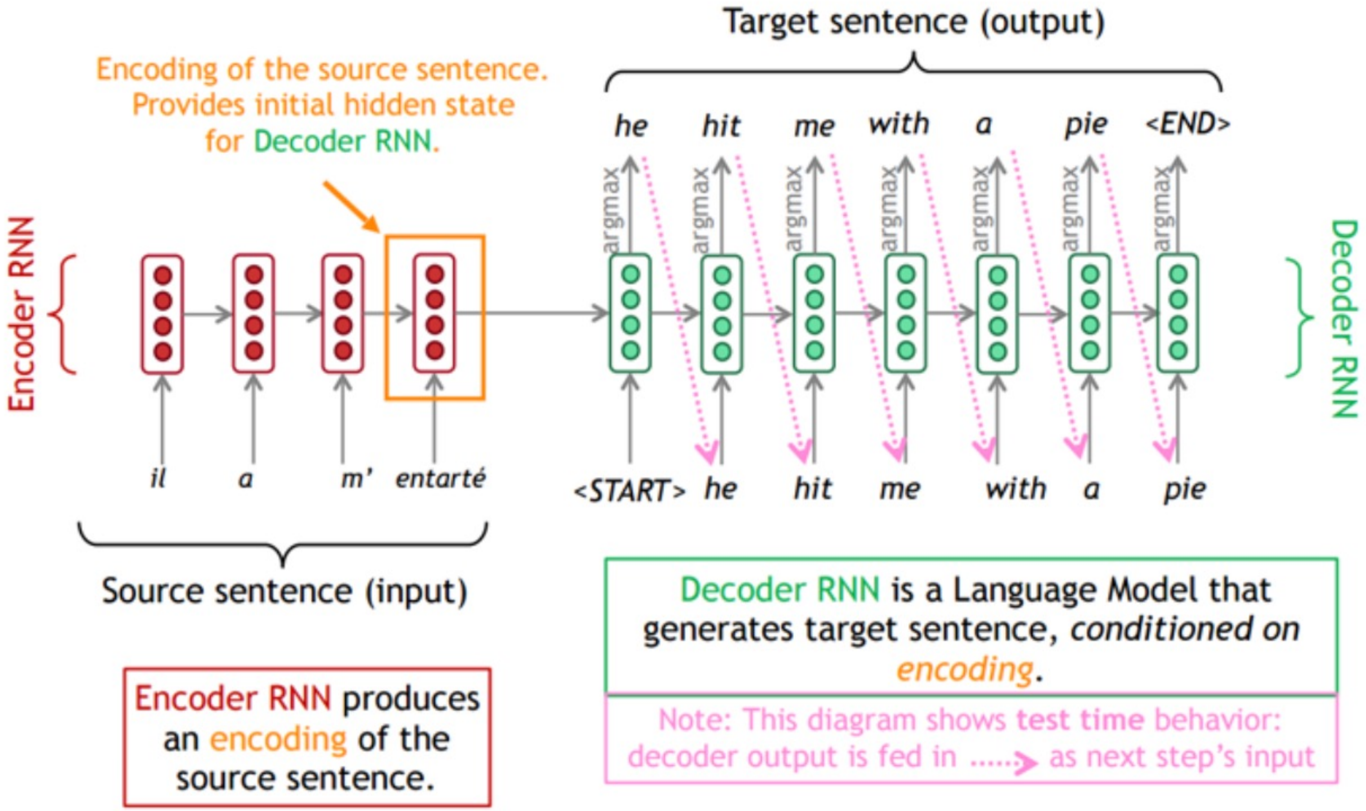


difficult to train the RNN (due to long term dependency) → **LSTM**

<EOS> symbol enables the model to define a distribution over ***sequences*** **of all possible lengths**

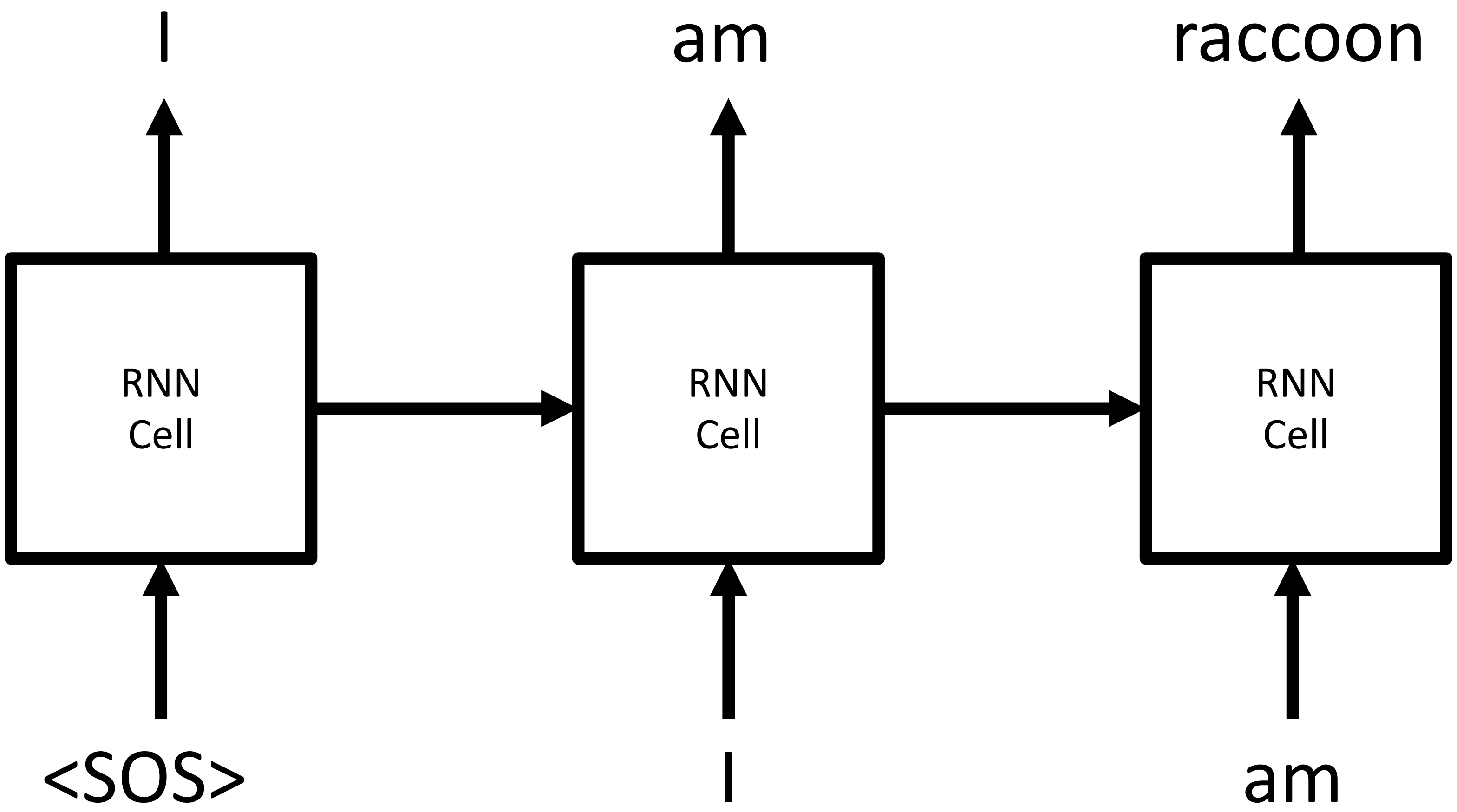
$$p(y_1, \dots, y_{T'} | x_1, \dots, x_T) = \prod_{t=1}^{T'} p(y_t | v, y_1, \dots, y_{t-1})$$

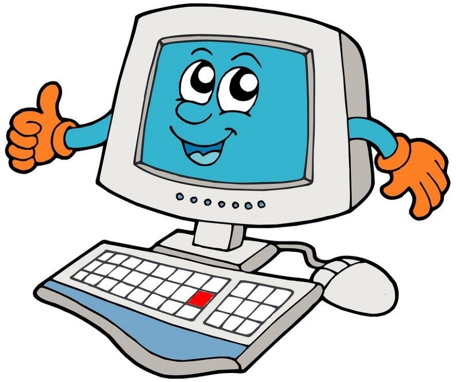
seq2seq



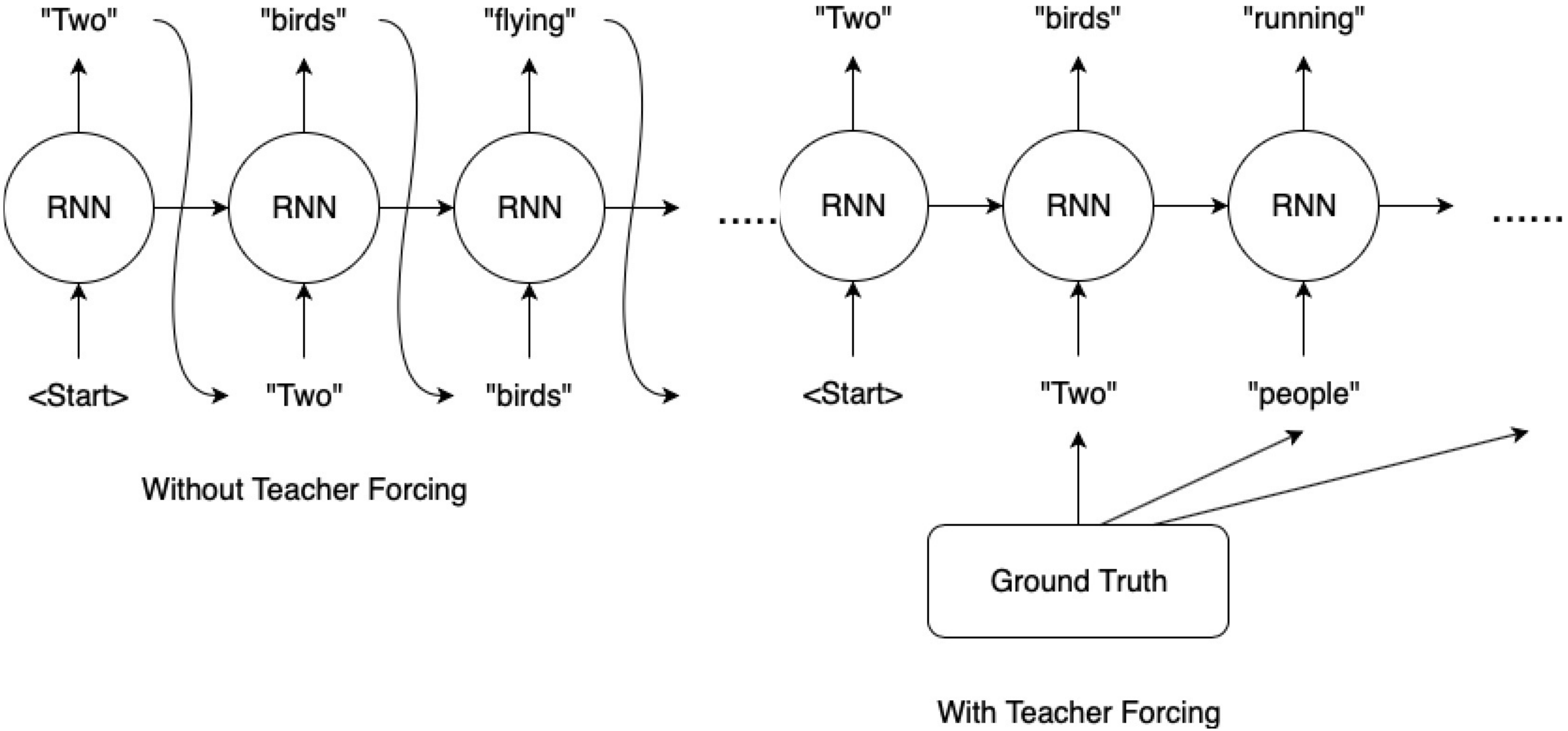
we want ***conditional probability* of output sequence** by given input sequence

Text Generation



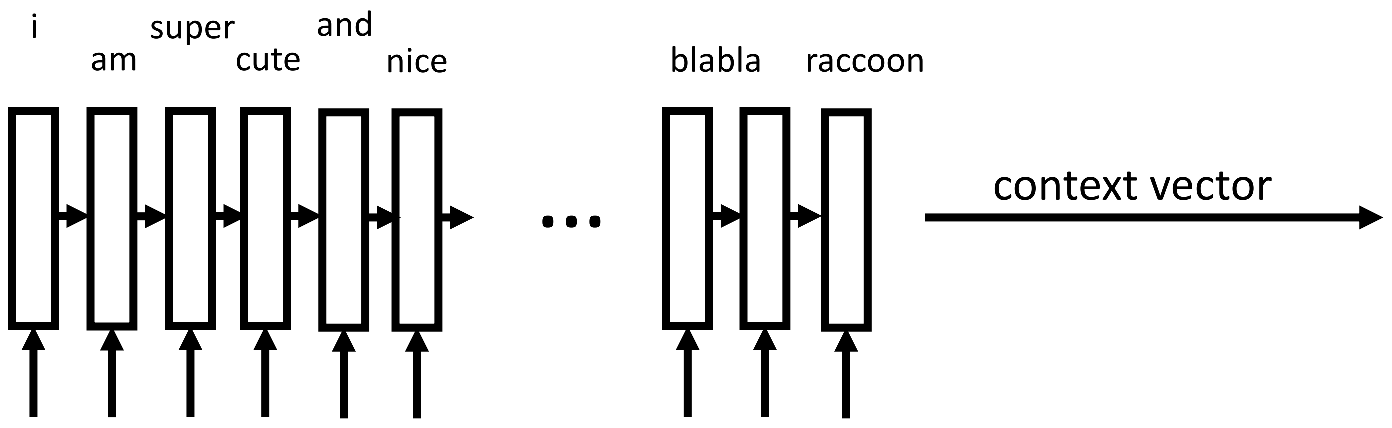


Teacher Forcing

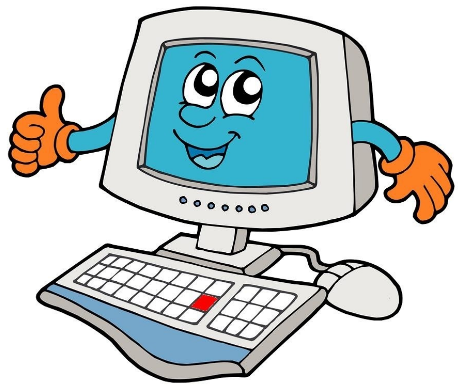


Code Review

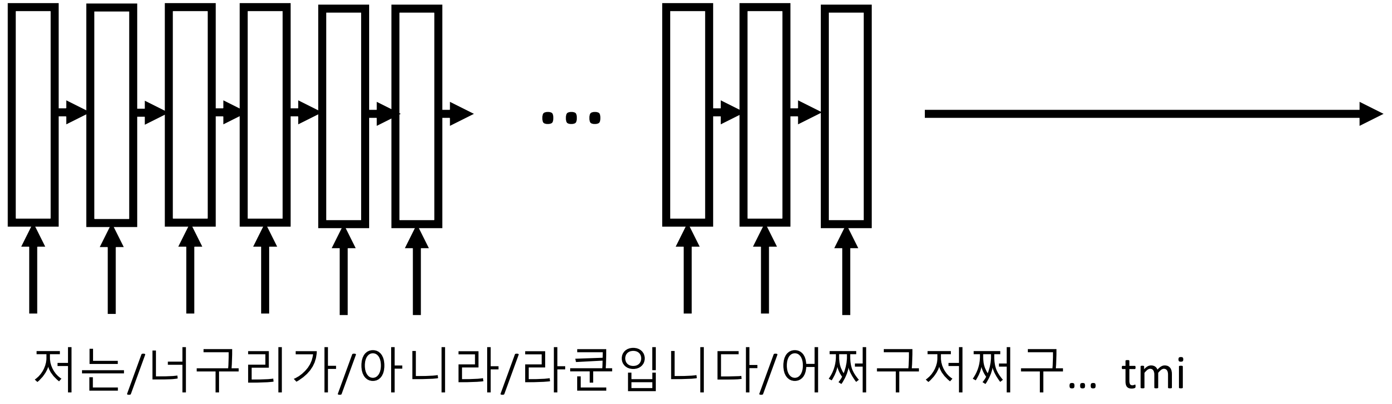
seq2seq



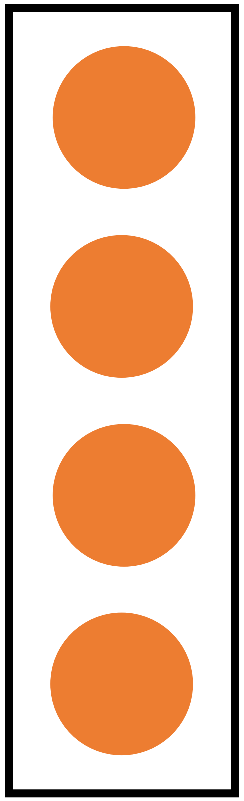
?

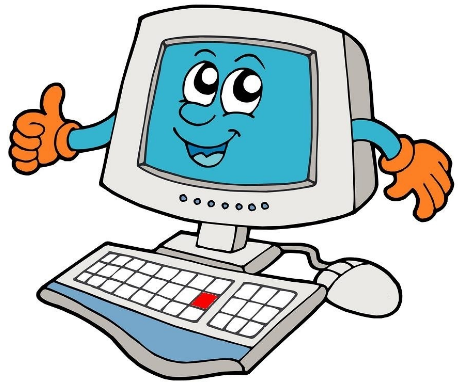


seq2seq



context vector
(4x1 vector)





Attention Visualizations

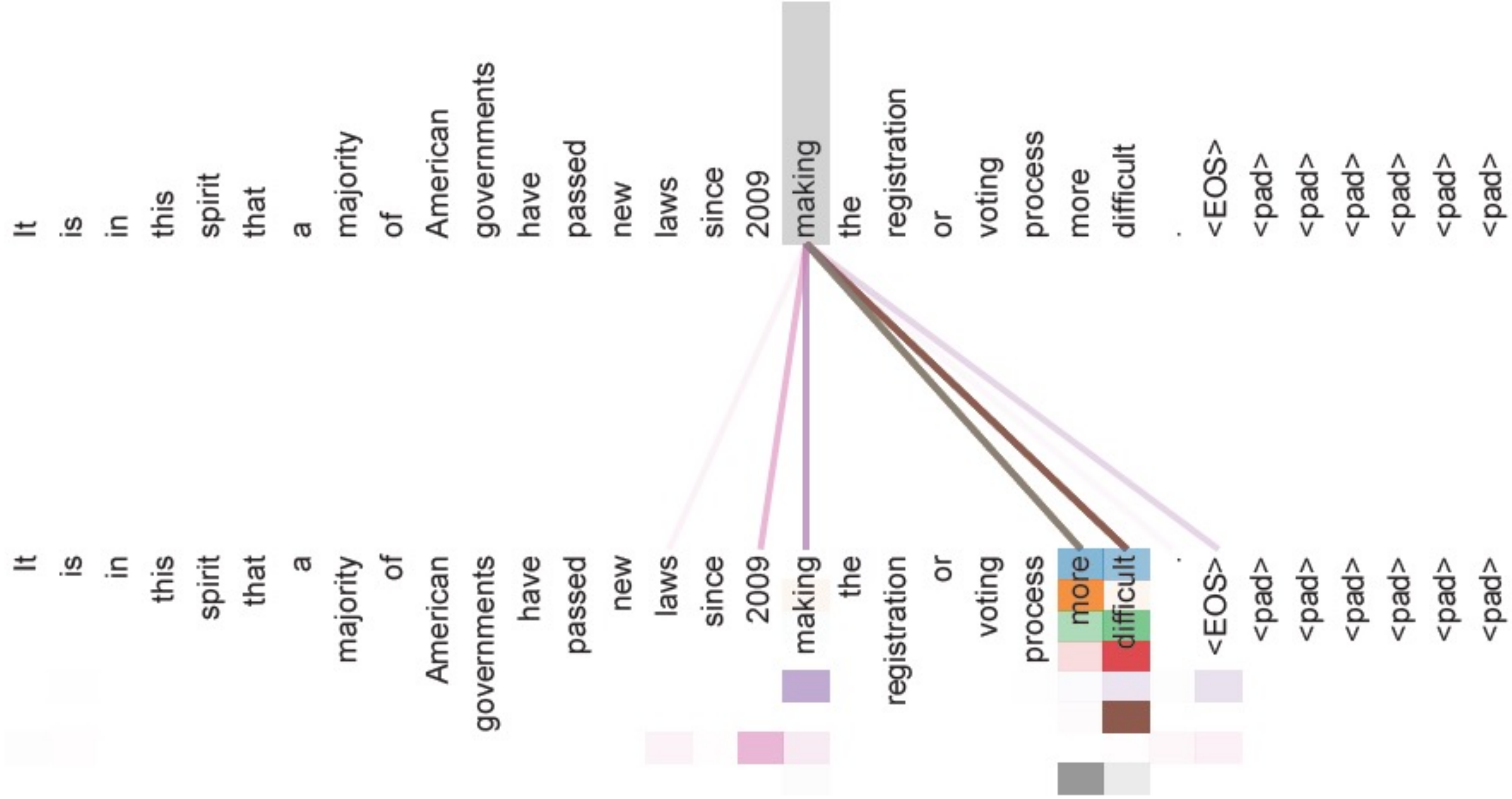


Figure 3: An example of the attention mechanism following long-distance dependencies in the encoder self-attention in layer 5 of 6. Many of the attention heads attend to a distant dependency of the verb ‘making’, completing the phrase ‘making...more difficult’. Attentions here shown only for the word ‘making’. Different colors represent different heads. Best viewed in color.

Neural Machine Translation by Jointly Learning to Align and Translate(2014, Bahdanau et al.)

$p(y_i|y_1, \ldots, y_{i-1}, x) = g(y_{i-1}, s_i, c_i)$

each conditional probability at decoder rnn cell

$s_i = f(s_{i-1}, y_{i-1}, c_i)$

s_i is an RNN hidden state for time i

$(h_1, \ldots, h_{T_x})$

encoder maps the input sentence into a sequence of annotations
context vector c_i depends on this.

★ ***each annotation h_i contains information about the whole input sequence with a strong focus on the parts surrounding the i_th word of the input sequence.***

Neural Machine Translation by Jointly Learning to Align and Translate(2014, Bahdanau et al.)

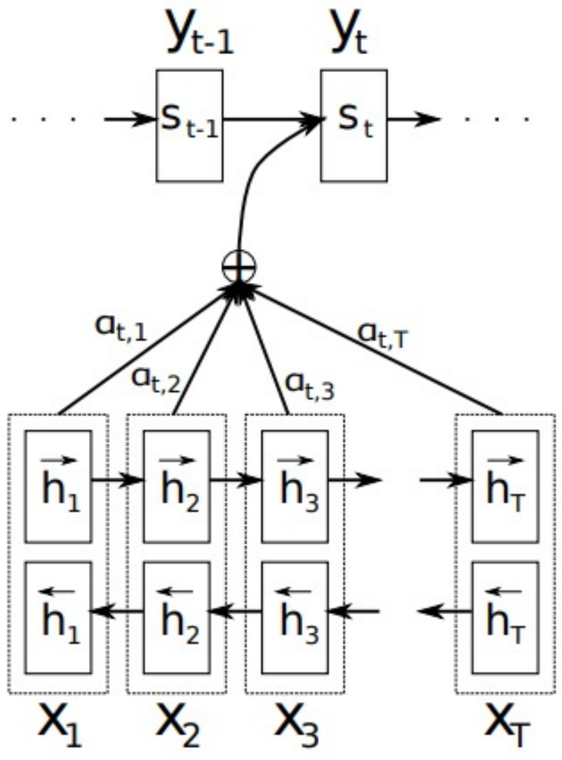


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

$$(h_1, \ldots, h_{T_x})$$

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

context vector c_i can be described as weighted sum of annotaions hi.

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})}$$

i = decoder i-th time step
j = input sequences’ j-th annotation

$$e_{ij} = a(s_{i-1}, h_j)$$

a is an alignment model.
scores how well the inputs around position j and the output at position i match.

Neural Machine Translation by Jointly Learning to Align and Translate(2014, Bahdanau et al.)

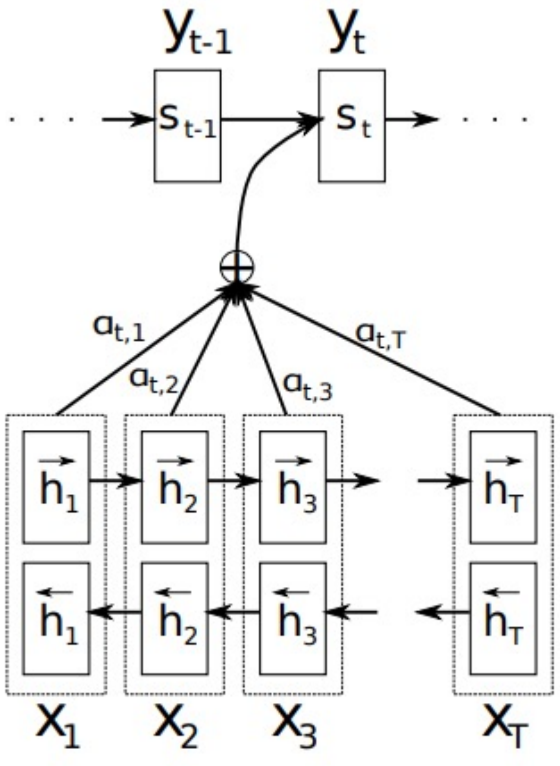


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

alignment model a

$$e_{ij} = a(s_{i-1}, h_j)$$

feed-forward neural network!

Neural Machine Translation by Jointly Learning to Align and Translate(2014, Bahdanau et al.)

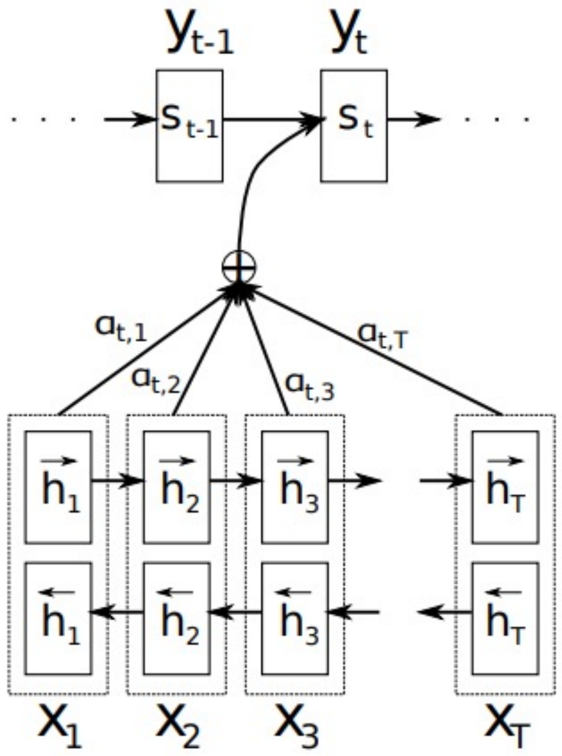


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

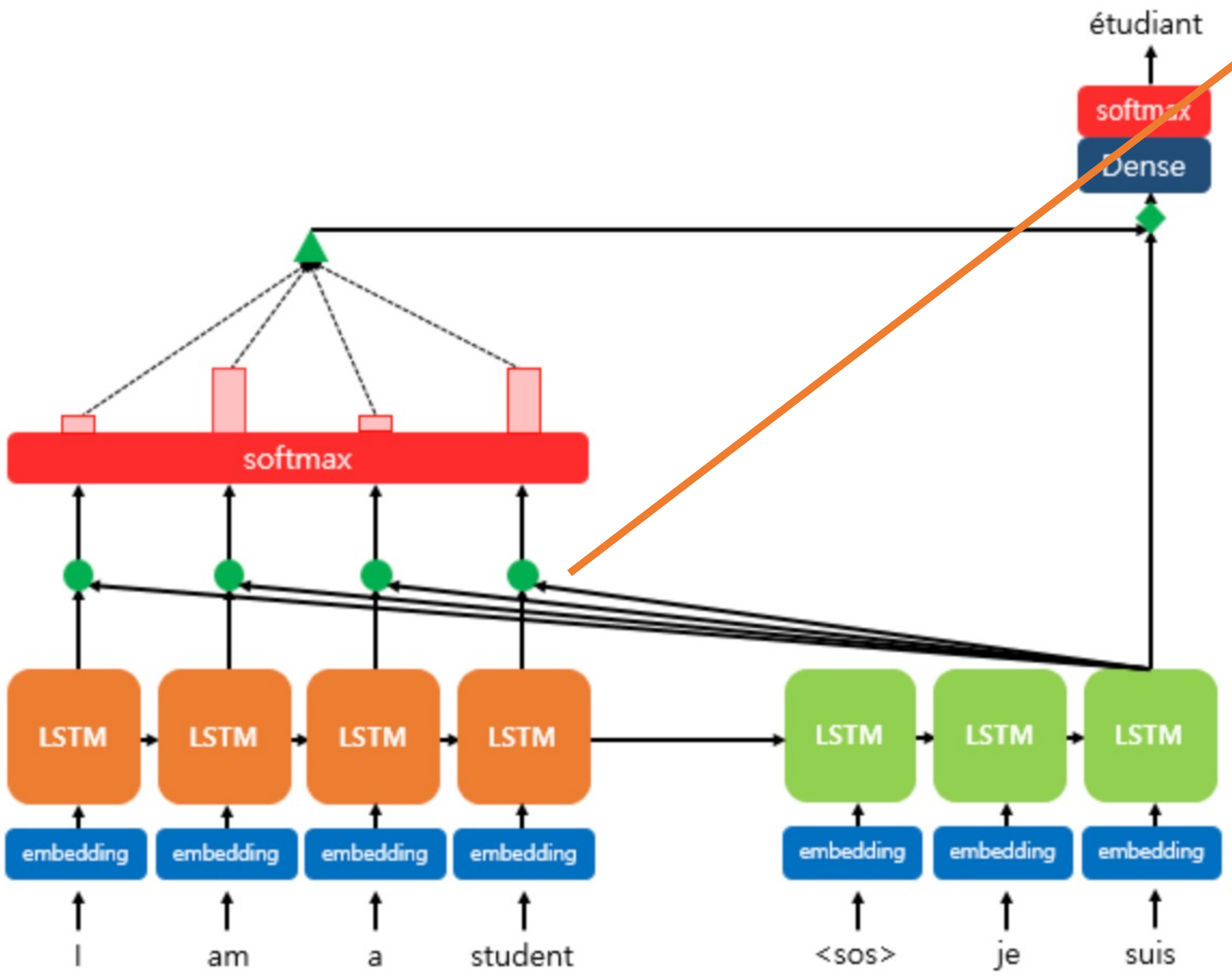
bi-directional RNN
→not only the proceding words, but also the following words.

$(\overrightarrow{h}_1, \cdots, \overrightarrow{h}_{T_x})$ forward hidden state

$(\overleftarrow{h}_1, \cdots, \overleftarrow{h}_{T_x})$ backward hidden state

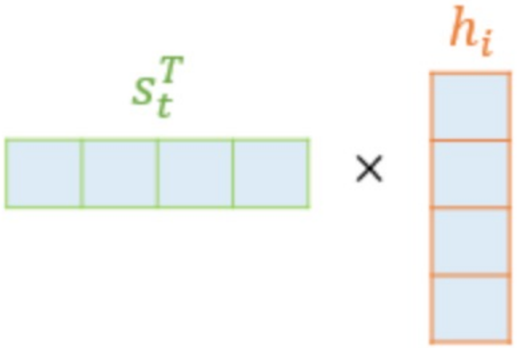
$h_j = \left[\overrightarrow{h}_j^\top; \overleftarrow{h}_j^\top\right]^\top$ concatenation

Attention Mechanism



$$e_{ij} = a(s_{i-1}, h_j)$$

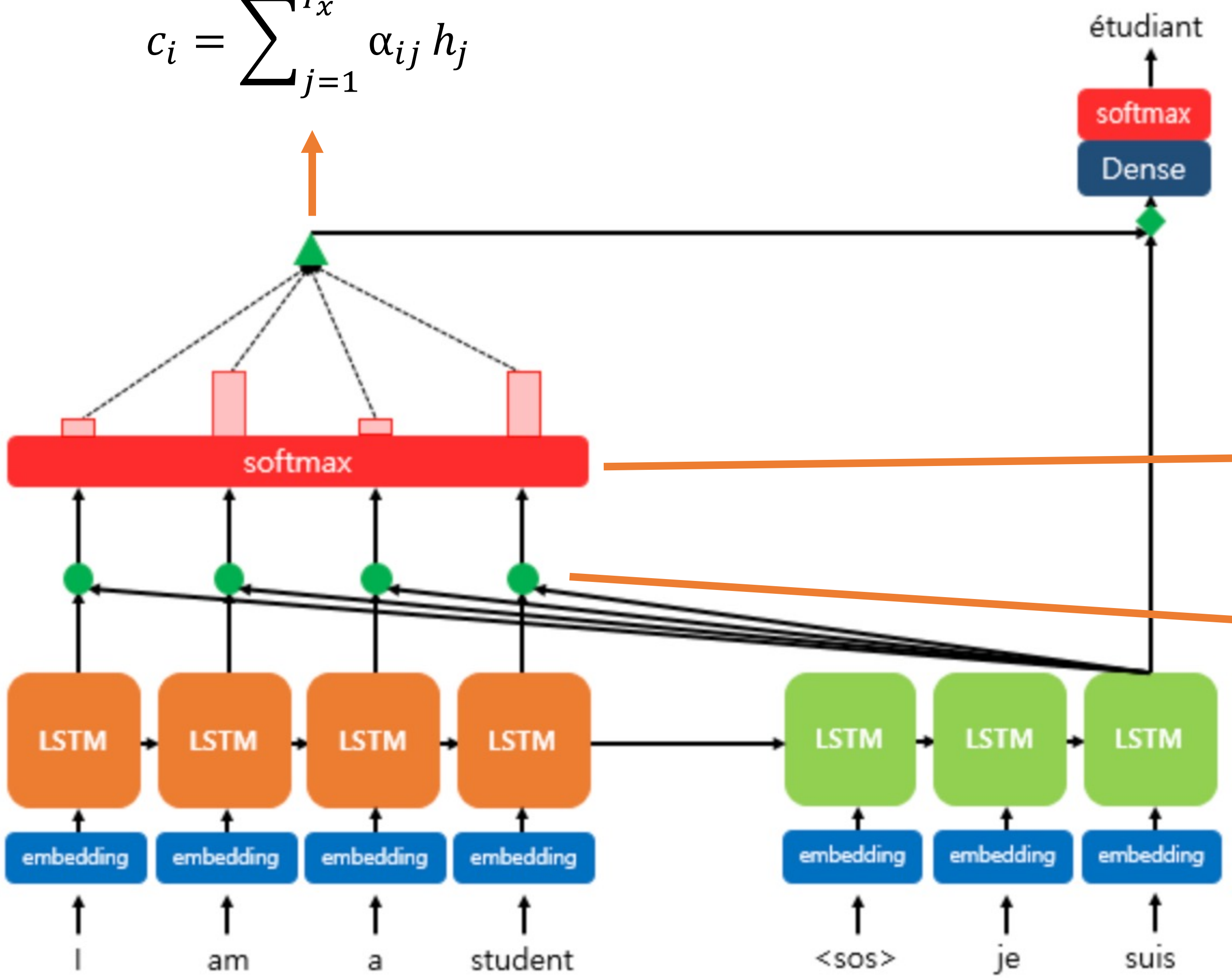
dot-product attention score



$$a(s_{i-1}, h_j) = s_{i-1}^T h_j$$

Attention Mechanism

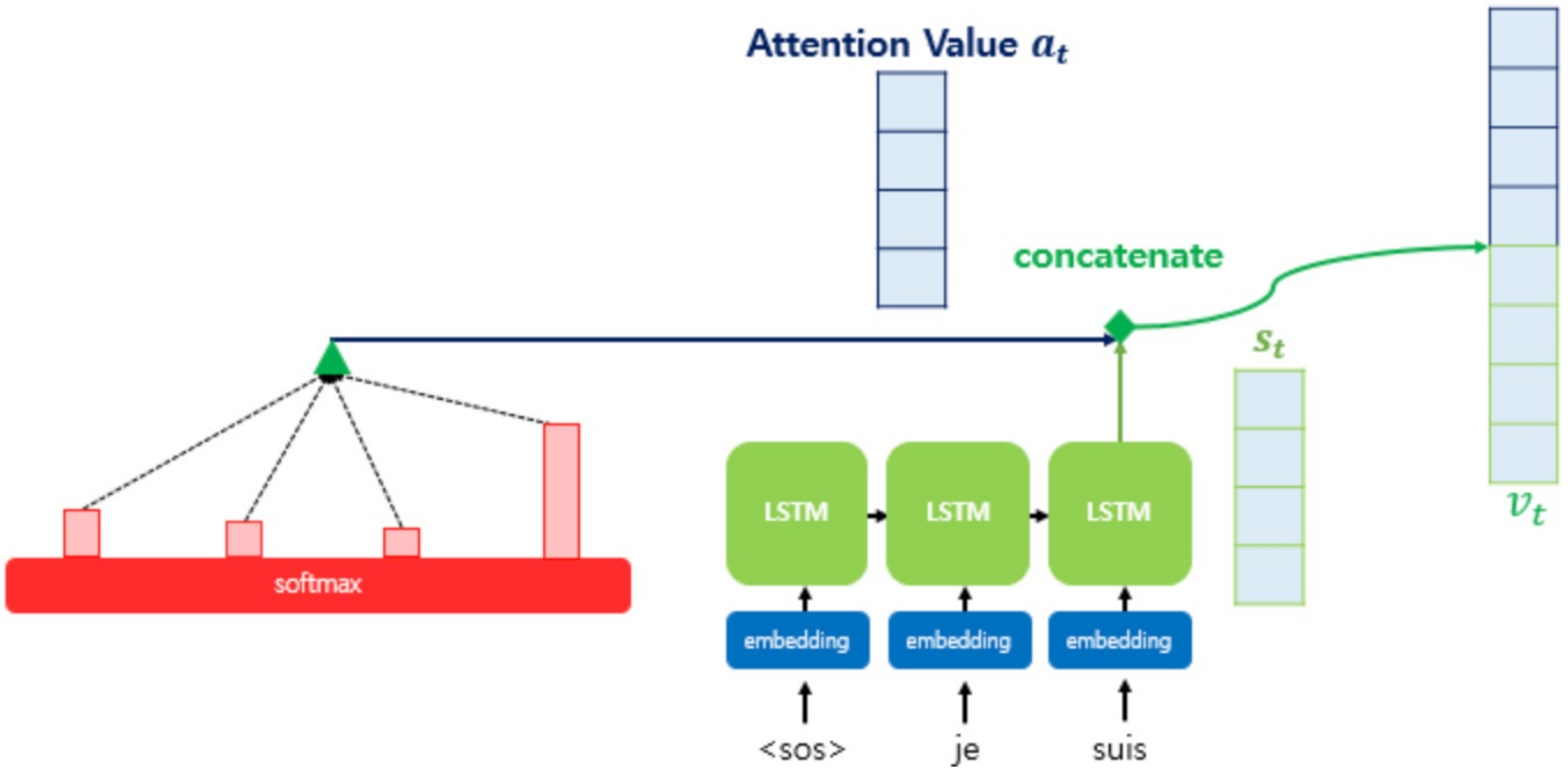
$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$



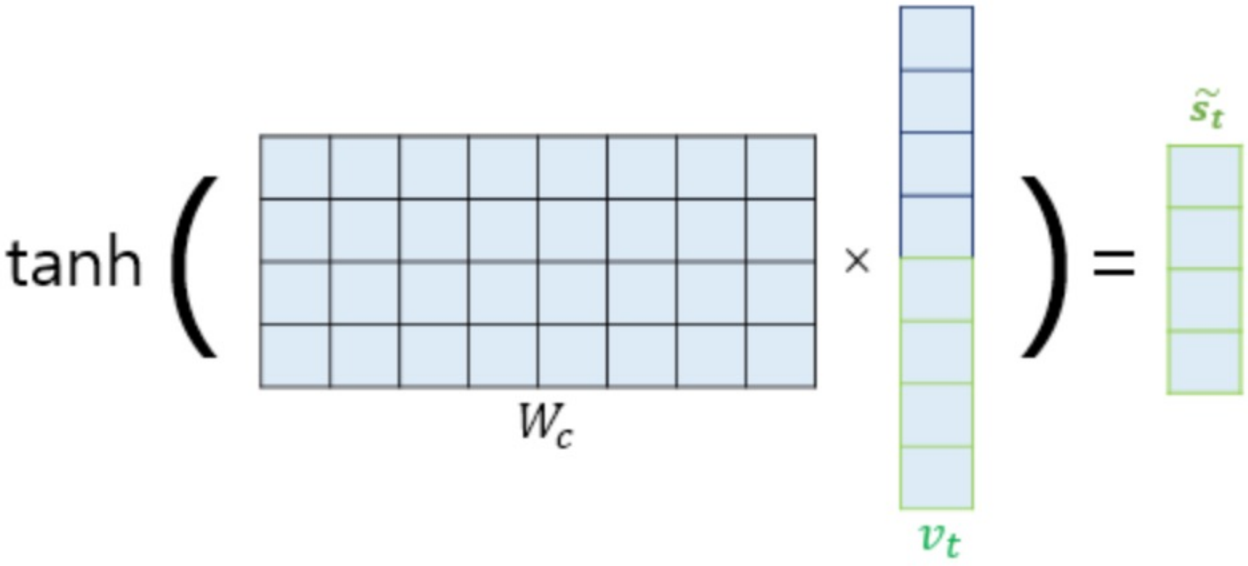
$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1}, h_j)$$

Attention Mechanism



Attention Mechanism



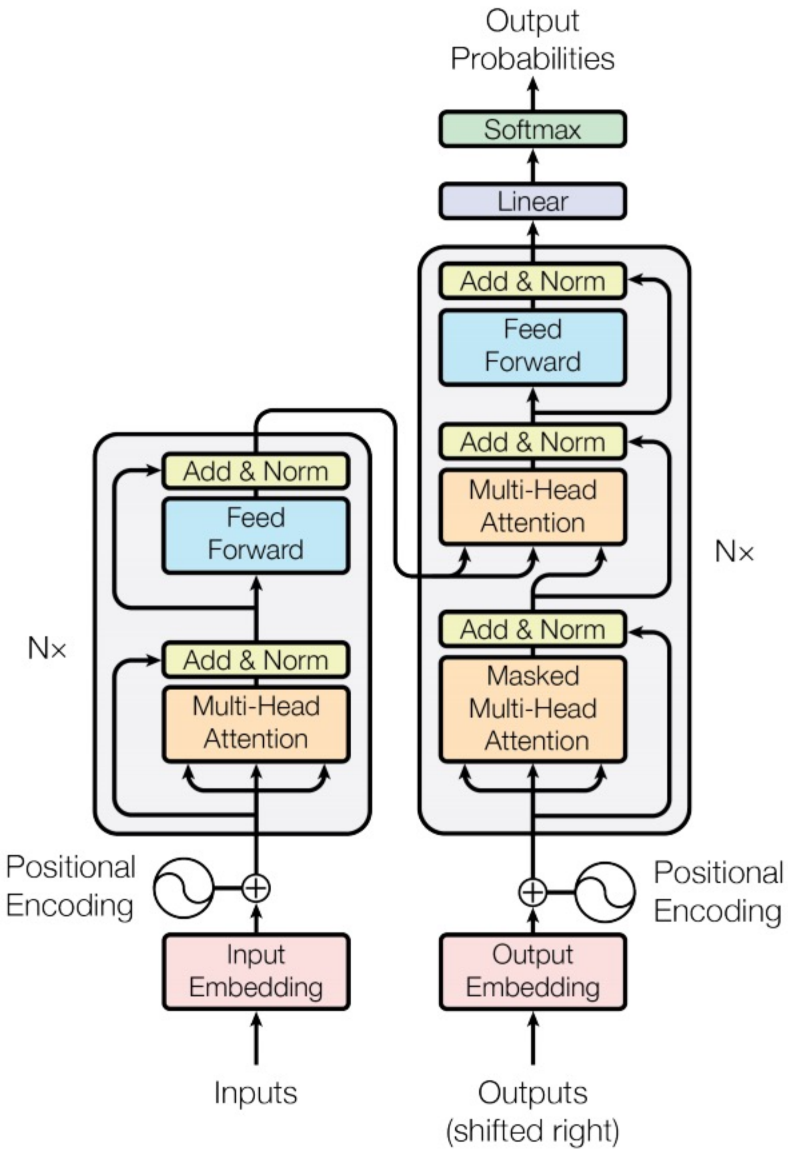
$$\tilde{s}_t = tanh(W_c v_t + b_c)$$

$$\hat{y}_t = Softmax(W_y \tilde{s}_t + b_y)$$

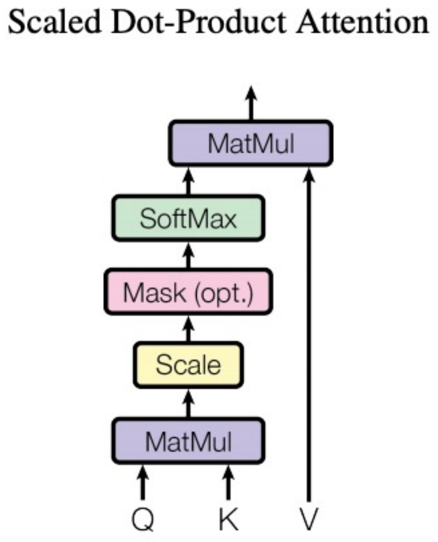
Attention Score function

이름	스코어 함수	Defined by
dot	$score(s_t,\ h_i) = s_t^T h_i$	Luong et al. (2015)
$scaled\ dot$	$score(s_t,\ h_i) = \frac{s_t^T h_i}{\sqrt{n}}$	Vaswani et al. (2017)
$general$	$score(s_t,\ h_i) = s_t^T W_a h_i$ // 단, W_a는 학습 가능한 가중치 행렬	Luong et al. (2015)
$concat$	$score(s_t,\ h_i) = W_a^T\ tanh(W_b[s_t; h_i])score(s_t,\ h_i) = W_a^T\ tanh(W_b s_t + W_c h_i)$	Bahdanau et al. (2015)
$location - base$	$\alpha_t = softmax(W_a s_t)$ // α_t 산출 시에 s_t만 사용하는 방법.	Luong et al. (2015)

Attention Is All You Need (2017, Vaswani et al.)



Attention Is All You Need (2017, Vaswani et al.)



Query: *decoder hidden state* at **time step t**

Keys: (**all**) *encoder cells' hidden state*

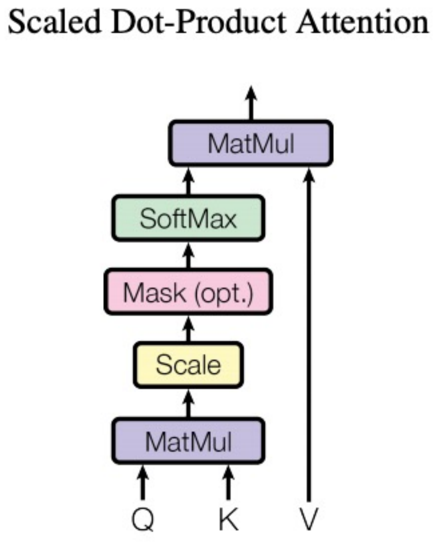
Values: (**all**) *encoder cells' hidden state*

$$score(s_t, h_i) = \frac{s_t^T h_i}{\sqrt{n}}$$

$$\alpha_{ij} = \frac{score(s_t, h_i)}{\sum_{k=1}^{T_x} score(s_t, h_k)}$$

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

Attention Is All You Need (2017, Vaswani et al.)



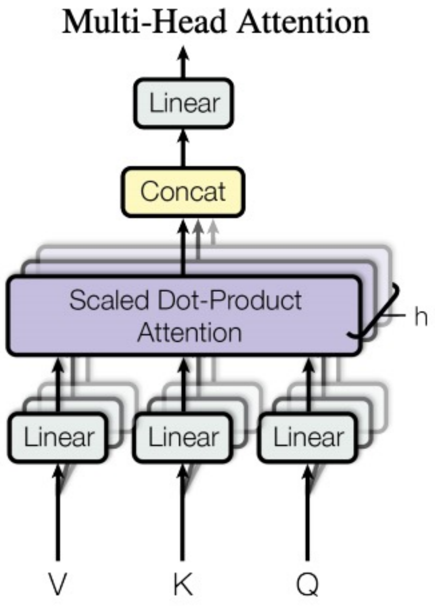
Query: *decoder hidden state* at **time step t**

Keys: (**all**) *encoder cells' hidden state*

Values: (**all**) *encoder cells' hidden state*

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

Attention Is All You Need (2017, Vaswani et al.)



$$Multihead(Q,K,V) = Concat(head_1, \dots, head_h)W^O$$

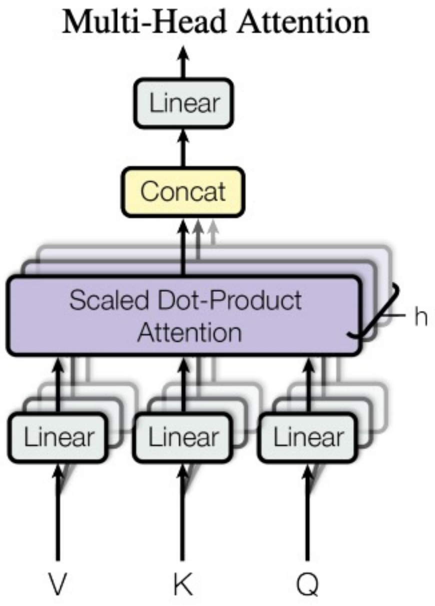
where $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$

$$W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$$

$$W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$$

$$d_k = d_v = d_{model}/h = 64$$

Attention Is All You Need (2017, Vaswani et al.)



$$QW_i^Q = [d_Q \times d_{model}] \times [d_{model} \times d_k] = [d_Q \times d_k]$$

$$KW_i^K = [d_K \times d_{model}] \times [d_{model} \times d_k] = [d_K \times d_k]$$

$$VW_i^V = [d_V \times d_{model}] \times [d_{model} \times d_v] = [d_V \times d_v]$$

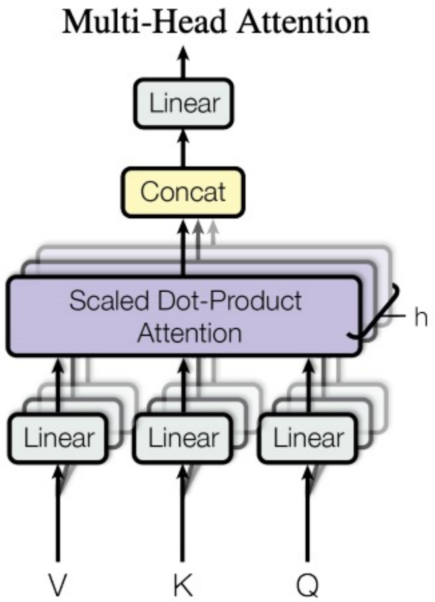
$$\downarrow$$

$$Attention(QW_i^Q, KW_i^K, VW_i^V) = [d_V \times d_v]$$

$$\downarrow$$

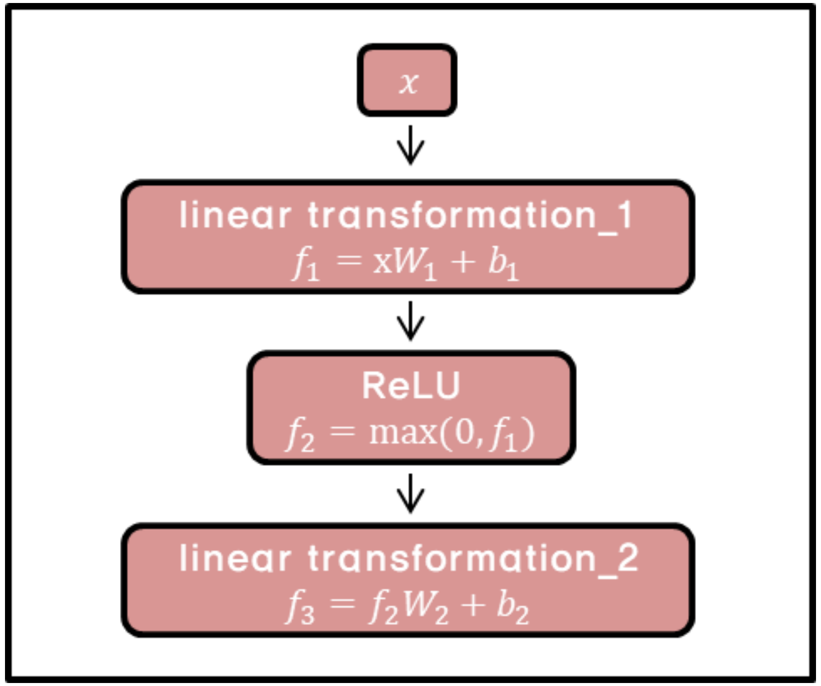
$$Concat(QW_i^Q, KW_i^K, VW_i^V)W^O = [d_V \times hd_v] \times [hd_v \times d_{model}] = [d_V \times d_{model}]$$

Attention Is All You Need (2017, Vaswani et al.)



Position-wise Feed-Forward Metworks

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



Attention Is All You Need (2017, Vaswani et al.)

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)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Attention Is All You Need (2017, Vaswani et al.)

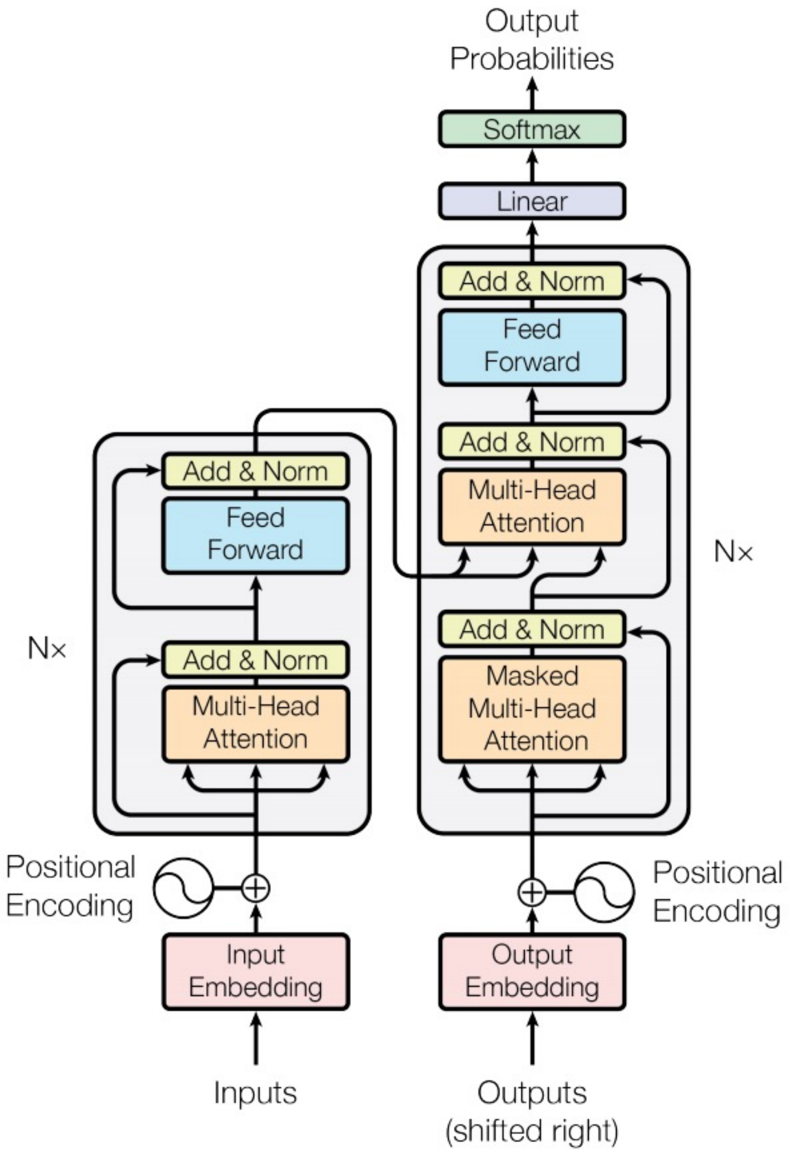
positional encoding

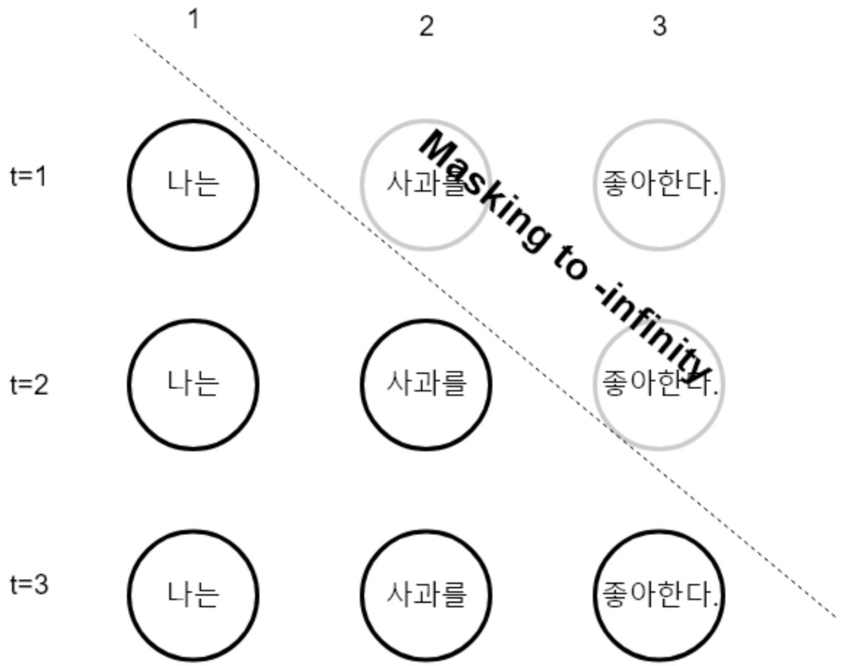
$$PE_{(pos_2 i)} = sin\left(po\,s/10000^{2i/d_{model}}\right)$$

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

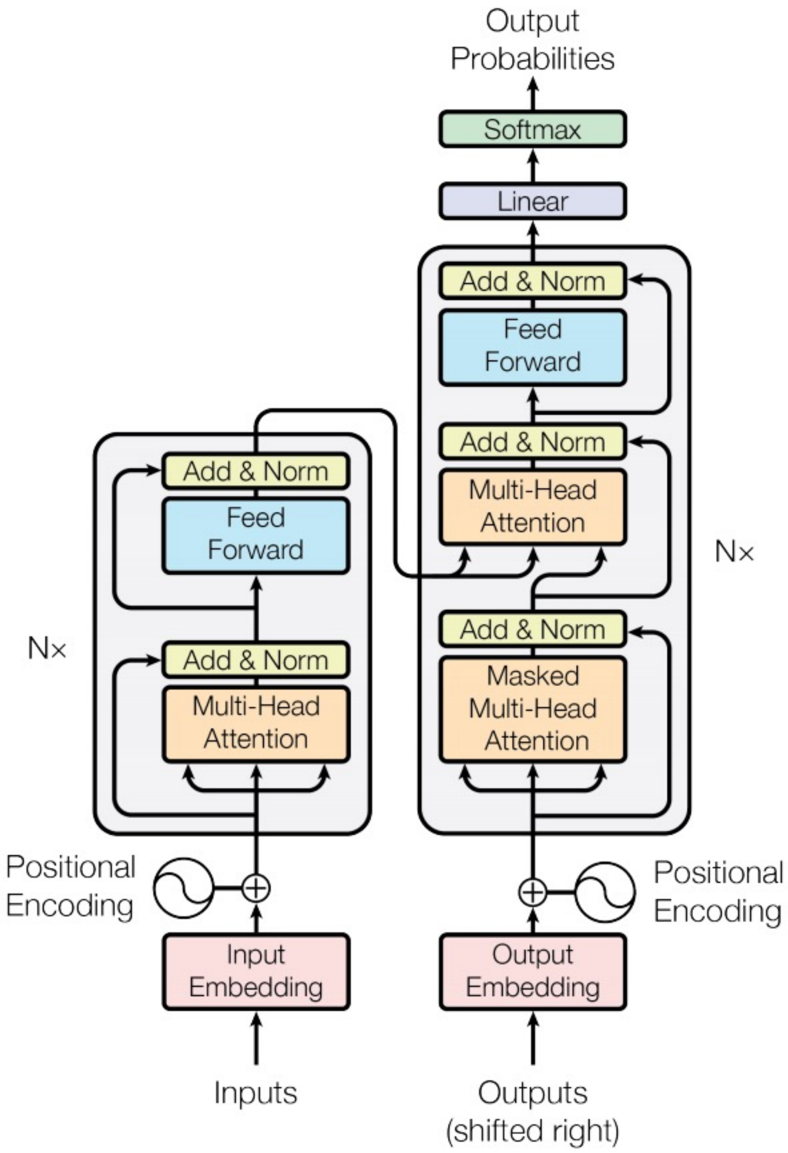
- allow the model to easily learn to attend by relative positions.
- PE_pos+k can be represented as a linear function of PE_pos.
- sinusoidal version allow the model to extraploate to sequence lengths longer than the ones encountered when training.

Attention Is All You Need (2017, Vaswani et al.)





Attention Is All You Need (2017, Vaswani et al.)



- no more long time dependency
- can emphasize values which are “close” to query
- parallelization (multi-head attention)

과제

- 예시 코드에서 데이터, rnn cell, attention 바꿔 가며 학습해 보기 (nmae score 비교하기)
- 다른 데이터 써도 됨
- scaled dot attention 구현