7주차 Review

5주차 : Text Classification; **Transformer Test 결과** / Trainer 구현

6주차 :「Attention is all you need」 논문 리뷰

7주차 : WMT 2016 **번역모델 구현**

2024.08.30

유하영

5주차 :

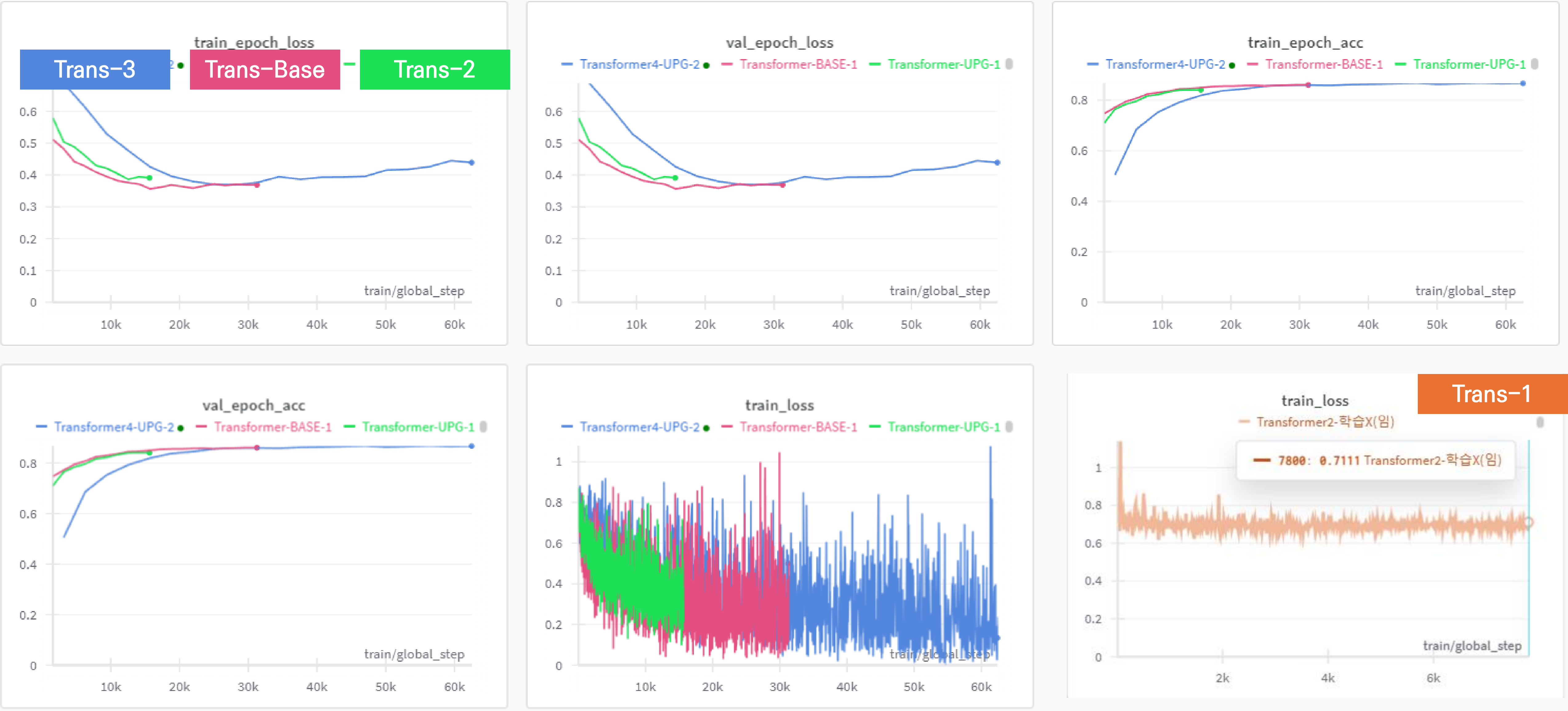
Text Classification; **Transformer Test 결과 확인**

[IMDB] Transformer Classification

Trans-Base	Trans-1	Trans-2	Trans-3
epoch = 20 lr = 2e-5 h = 8 N = 6 (layer)	epoch = 20 lr = 2e-5 h = 8 N = 6 (layer)	epoch = 10 lr = 2e-5 h = 8 N = 6 (layer)	epoch = 20 lr = 2e-5 h = 8 N = 6 (layer)
-	• 사전 학습된 임베딩 벡터 사용 (BERT) -〉 학습진행 X	• epoch 10으로 조정 • 모델 구조에 Dropout추가 - Encoder에서 출력된 임베딩에 드롭아웃 레이어 0.3 추가 • Data augmentation (random_deletion, random_swap) 0.2 확률로 적용, 2배 증강	Trans-2에 추가 적용 • batch_size= 16 → 8 • 학습률이 10%동안 0에서 점진적으로 증가 → 2e-5 사용

```
Original train size :25000
Augmented train size :50000
```

[IMDB] Transformer Classification



[IMDB] Transformer Classification

Train/val

	Trans-Base	Trans-1	Trans-2	Trans-3
Train_loss	0.3117	X	0.2712	0.1831
Val_Loss	**0.3689**	X	0.3911	0.4390
Val_acc	0.8607	X	0.8411	**0.8672**

Test

	Trans-Base	Trans-1	Trans-2	Trans-3
Accuracy	0.86072	X	0.84116	**0.86728**
Loss	**0.36898**	X	0.39110	0.43902

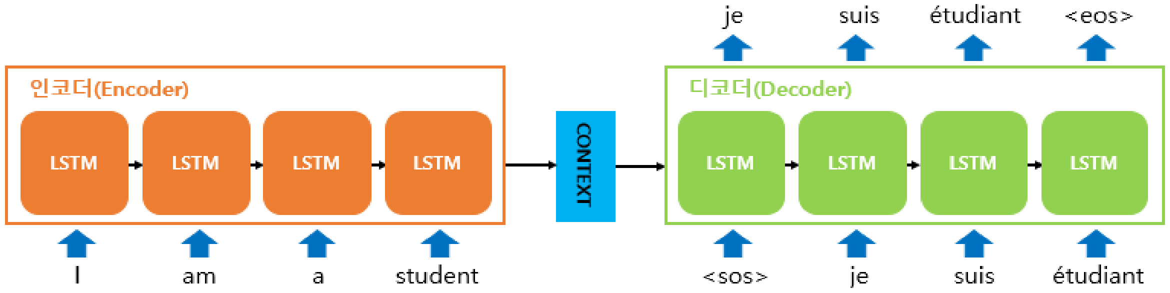
Attention is All you need

Vaswani, Ashish, et al. "Attention is all you need."
Advances in neural information processing systems 30 (2017).

Background

단일 Attention mechanism을 사용한 Transformer 모델을 제안

RNN -〉 seq2seq -〉 seq2seq with Attention



Self-Attention

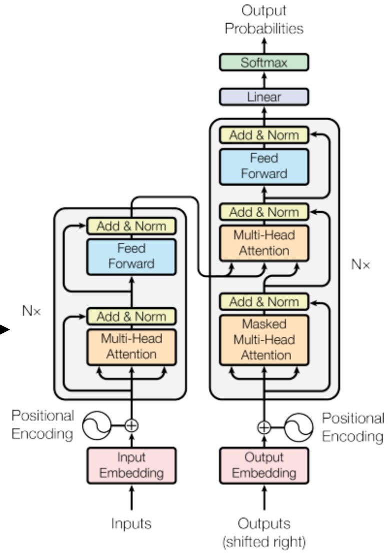


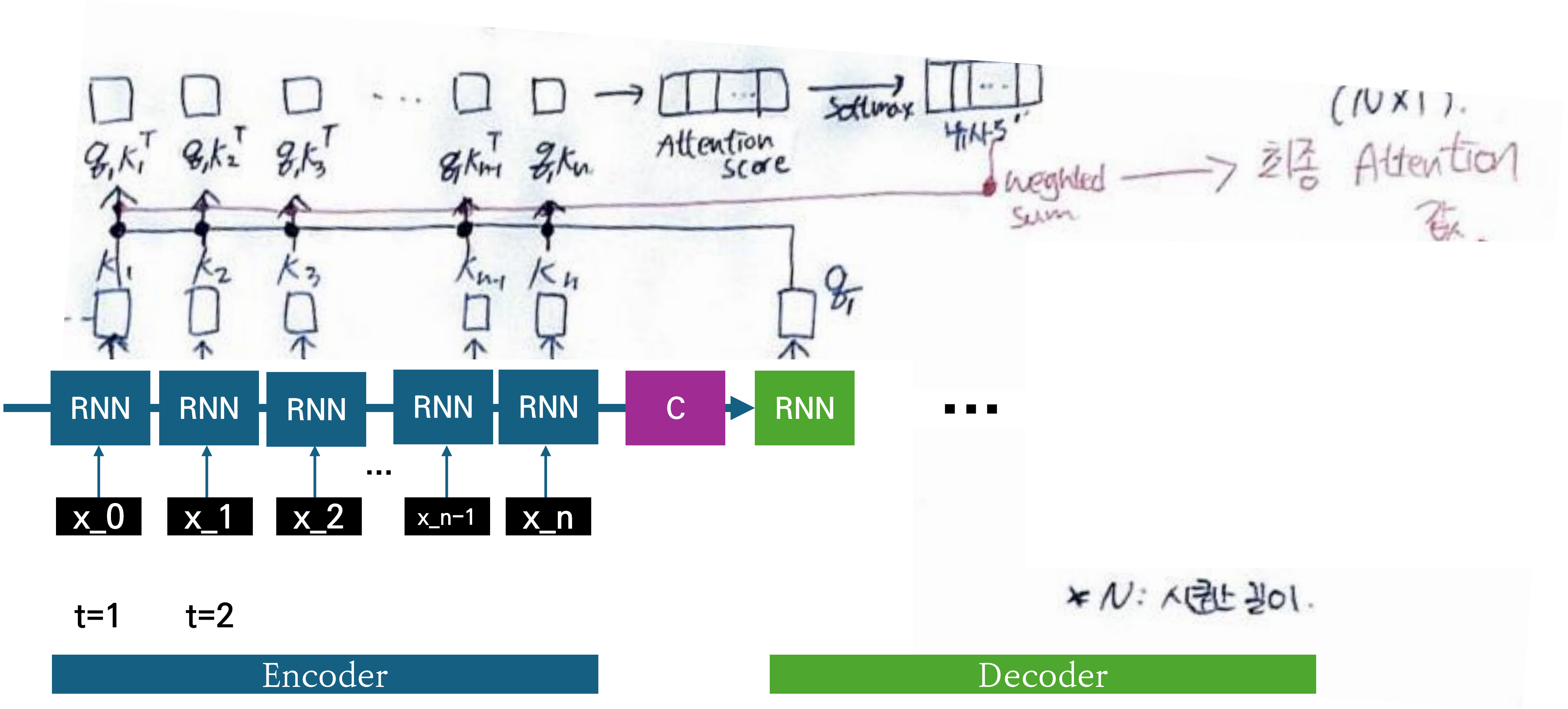
Figure 1: The Transformer - model architecture.

Introduction
Background
Model Architecture
3.1 Encoder and Decoder Stacks
3.2 Attention
Scaled Dot-Product Attention
Multi-Head Attention
Applications of Attention in our Model
3.3 Position-wise Feed-Forward Networks
3.4 Embeddings and Softmax
3.5 Positional Encoding
Why Self-Attention?
Results

- long sequence -〉 정보손실↑
- 병렬처리 불가

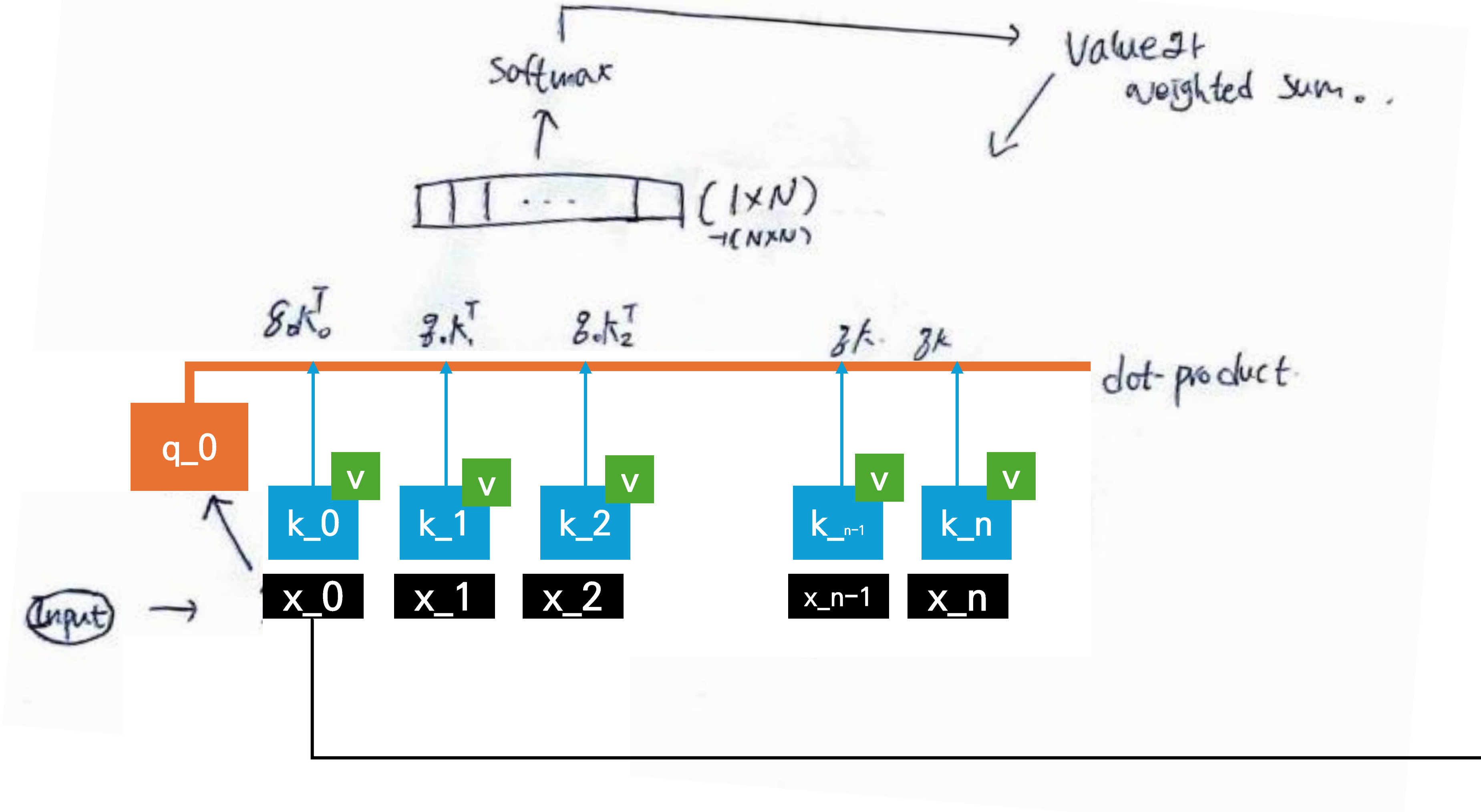
Self-Attention

* Attention

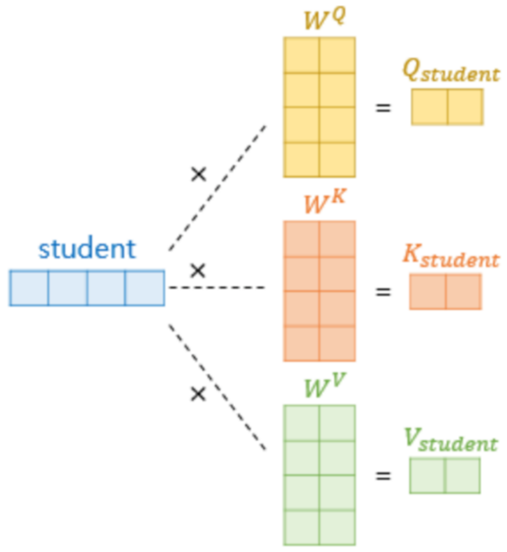


Self-Attention

* Self Attention



*q k v ?



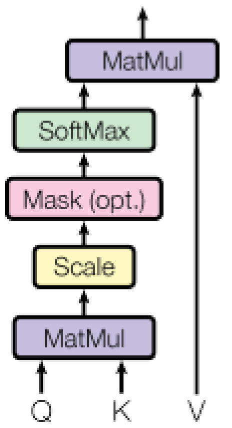
Self-Attention

* scaled Dot−Product Attention

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

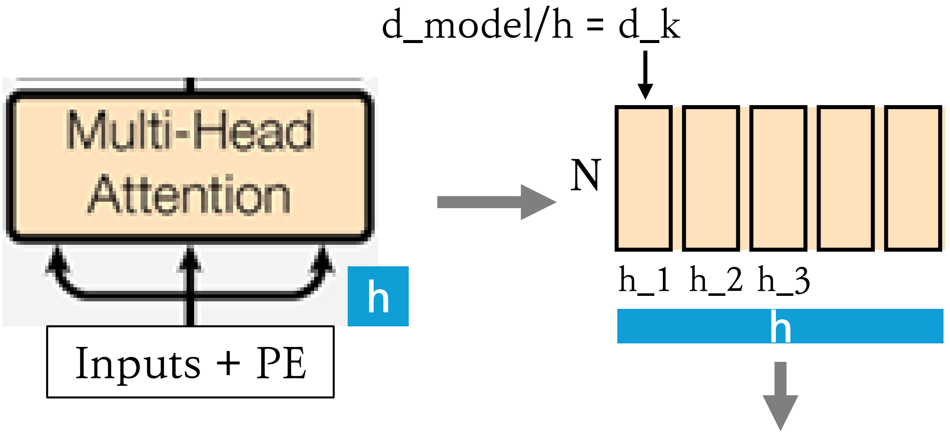
d_k 값이 클수록 dot product의 크기가 커져
소프트맥스 함수가 매우 작은 그래디언트를 갖게 되는 것을 방지

Scaled Dot-Product Attention



* Multi−Head Attention

$$\mathrm{MultiHead}(Q,K,V)=\mathrm{Concat}(\mathrm{head_1},...,\mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i}=\mathrm{Attention}(QW_i^Q,KW_i^K,VW_i^V)$$

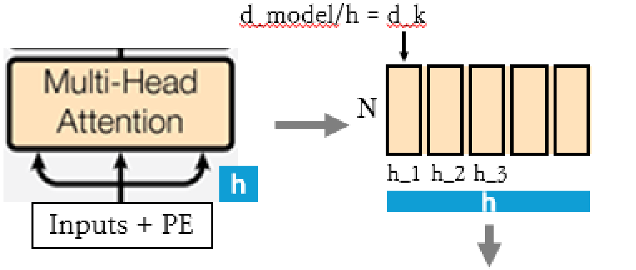


$$\mathrm{Concat}(\mathrm{head_1},...,\mathrm{head_h})W^O$$
$$= \mathrm{MultiHead}(Q,K,V)$$

Self-Attention

* Multi-Head Attention

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head}_1, ..., \mathrm{head}_h)W^O$$
$$\text{where } \mathrm{head}_i = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$



$$\mathrm{Concat}(\mathrm{head}_1, ..., \mathrm{head}_h)W^O$$
$$= \mathrm{MultiHead}(Q, K, V)$$

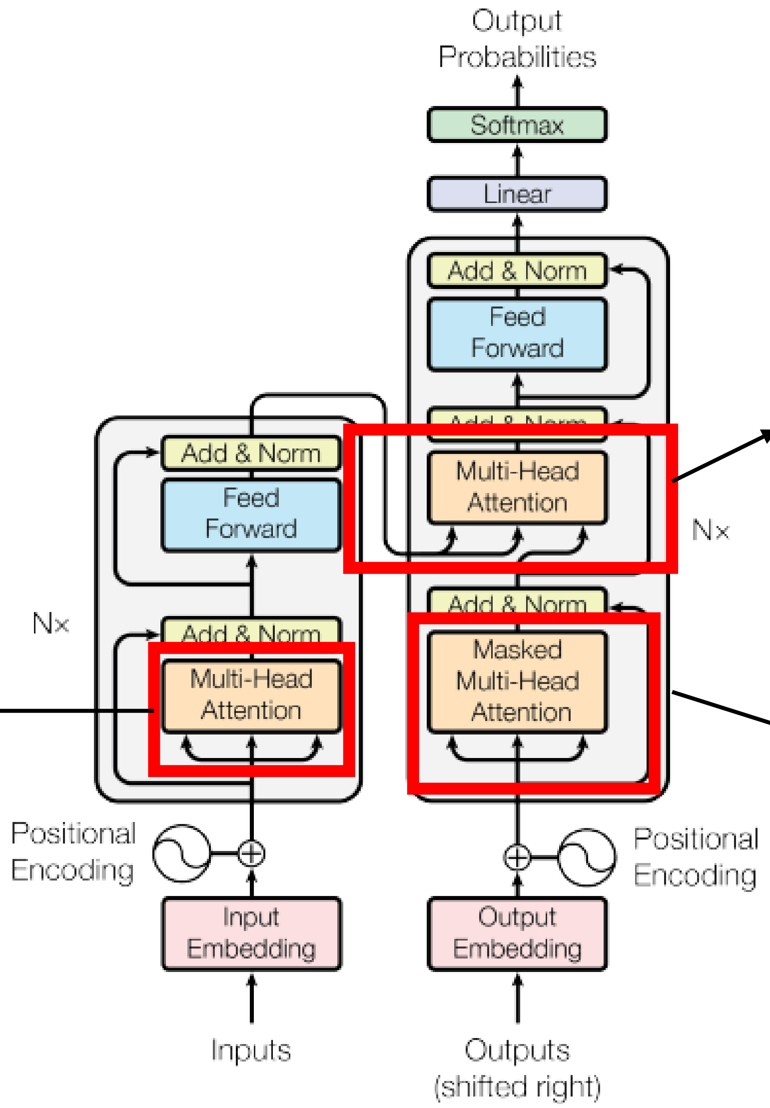
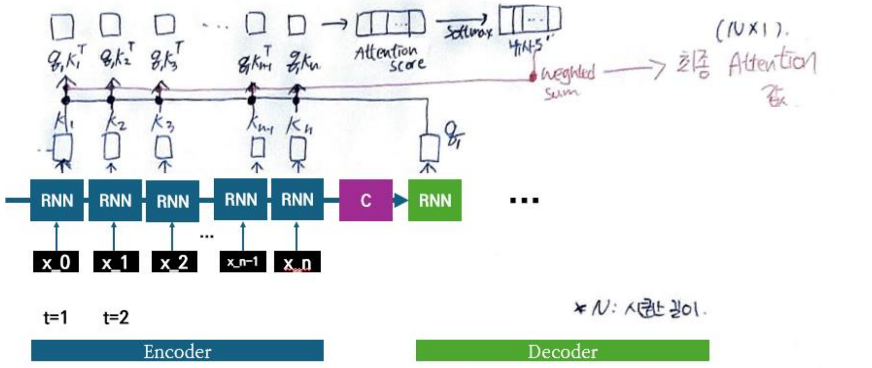
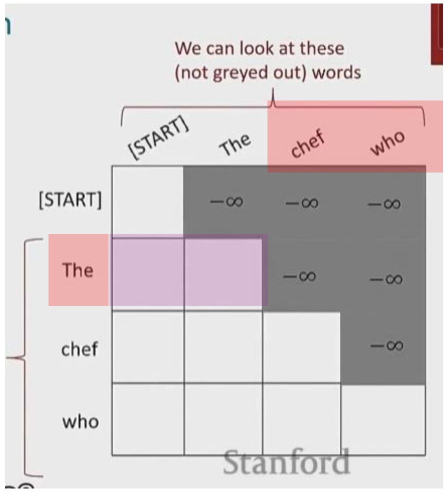


Figure 1: The Transformer - model architecture.

* seq2seq with Attention와 유사



* Masked Multi-head Attention



X

Model Architecture

* Positional Encoding

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

- 위치 정보의 고유성(다양한 인코딩이 생성)/상대적 위치 인식
- 특정 위치(예: 첫 번째 단어)에 대한 포지셔널 인코딩 값은 해당 위치에서 동일하게 유지
- 원래 임베딩 벡터의 내용적 의미를 크게 해치지 않는다.

* Residual Connection

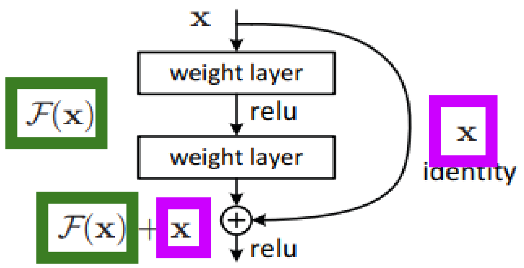


Figure 2. Residual learning: a building block.

$$H(x) = x + Multi-head\ Attention(x)$$

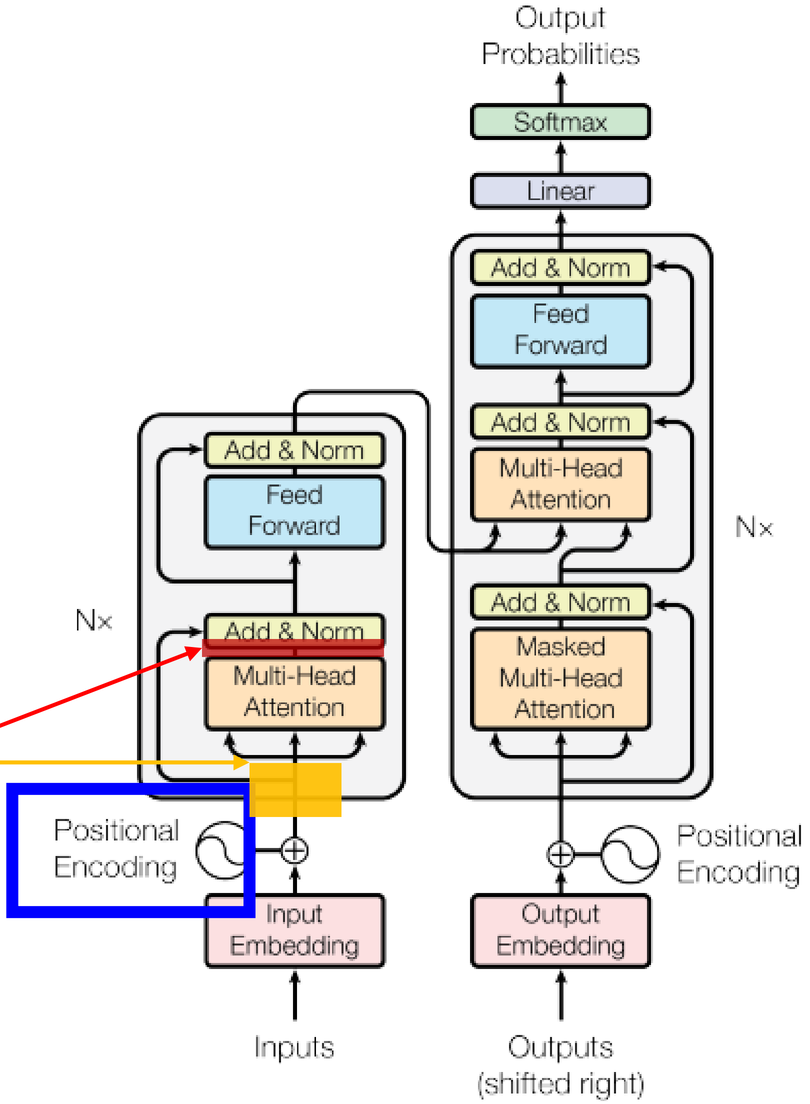


Figure 1: The Transformer - model architecture.

Model Architecture

* Position-wise Feed Forward Networks

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

각 단어의 벡터에 대해 개별적으로 계산

Feed Forward → 비선형성 추가).

① 차원확장

$FFN_1(x_i)$

$= W_1 x_i + b$

ReLU

$RELU(FFN_1(x_i))$

$= \max(0, W_1 x_i + b_1)$

② 차원축소(원래)

$FFN_2(\;)$

$= W_2 \cdot \underline{RELU(W_1 x_i + b_1)} + b_2$

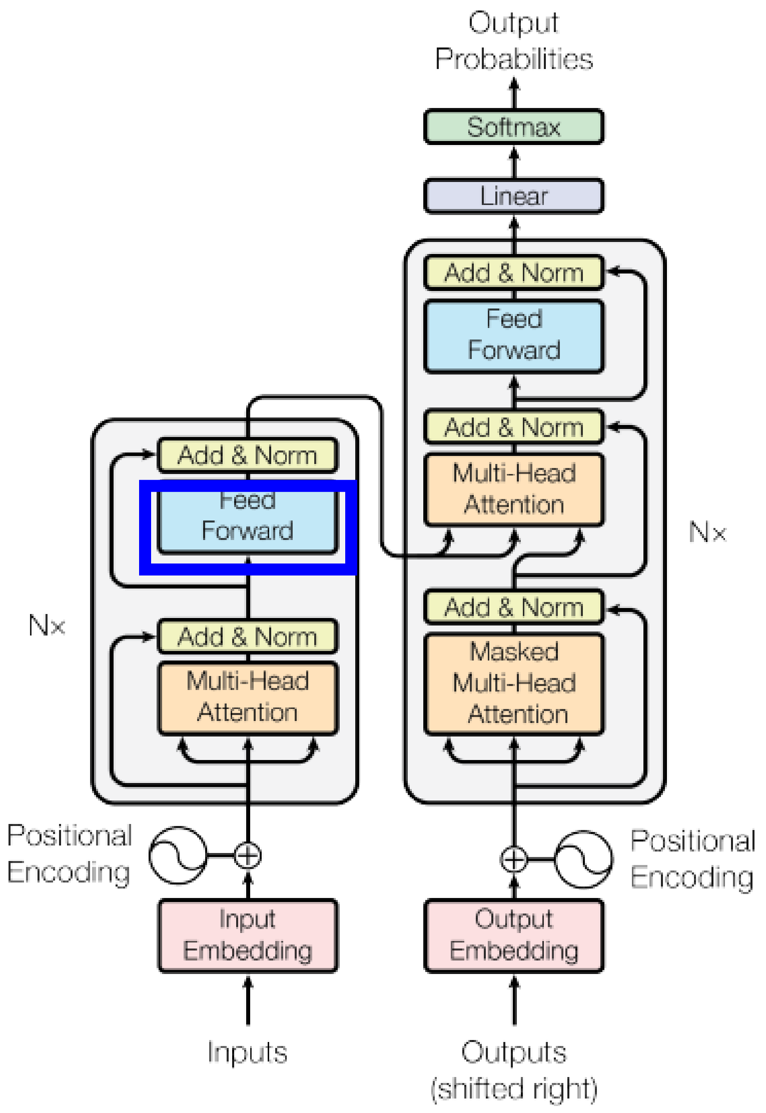


Figure 1: The Transformer - model architecture.

Results

WMT 2014

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 EN-DE	BLEU EN-FR	Training Cost (FLOPs) EN-DE	Training Cost (FLOPs) 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**	$2.3 \cdot 10^{19}$	

newstest2013 . English-to-German translation

Table 3: Variations on the Transformer architecture.

	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

7주차 :

WMT 2016 번역모델 구현

Transformer를 활용한 번역 모델 구현

1. 제한된 자원/메모리 부족

```
OutOfMemoryError: CUDA out of memory. Tried to allocate 5.88 GiB. GPU 0 has a total capacity of 14.74 GiB of which 1.22 GiB is free. Process 2975 has 13.52 GiB memory in use. Of the allocated memory 8.20 GiB is allocated by PyTorch, and 5.12 GiB is reserved by PyTorch but unallocated. If reserved but unallocated memory is large try setting PYTORCH_CUDA_ALLOC_CONF=expandable_segments:True to avoid fragmentation.  See documentation for Memory Management  (https://pytorch.org/docs/stable/notes/cuda.html#environment-variables)
```

2. 텐서 변환.. 문제..

Trainer X -> 텐서 변환 문제

Trainer O -> 컴퓨팅 자원 문제, 텐서 변환 문제

huggingface 라이브러리 -> 자원문제