Longformer : The Long-Document Transformer

INDEX

1. Longformer Scratch

01
02
03
04
05

❖ Propose long document NLP task.

- Longformer is BERT based model.
- Transformer based models are unable to process long sequence due to self-attention operation.
  - ·Transformer's computation scales quadratically with sequence length.
- Long document process in BERT :
  - · BERT cut long documents into small sequences.
  - · BERT has 512 token limit. Such partitioning could potentially result in loss of important cross-partition information, and to mitigate this problem, existing methods often rely on complex architectures to address such interactions.
  - · Example) Harry Potter is a series of seven fantasy novels. ··· ··· ··· he ···

< Relation : Is 'he' Harry Potter? Information Loss >

- Longformer's attention mechanism
  - · Longformer is able to build contextual representations of the entire context using multiple layers of attention making it easy to process documents of thousands of tokens or longer, also reducing computation and complexity of model.

Model	attention matrix	char-LM	other tasks	pretrain
Transformer-XL (2019)	ltr	yes	no	no
Adaptive Span (2019)	ltr	yes	no	no
Compressive (2020)	ltr	yes	no	no
Reformer (2020)	sparse	yes	no	no
Sparse (2019)	sparse	yes	no	no
Routing (2020)	sparse	yes	no	no
BP-Transformer (2019)	sparse	yes	MT	no
Blockwise (2019)	sparse	no	QA	yes
Our Longformer	sparse	yes	multiple	yes

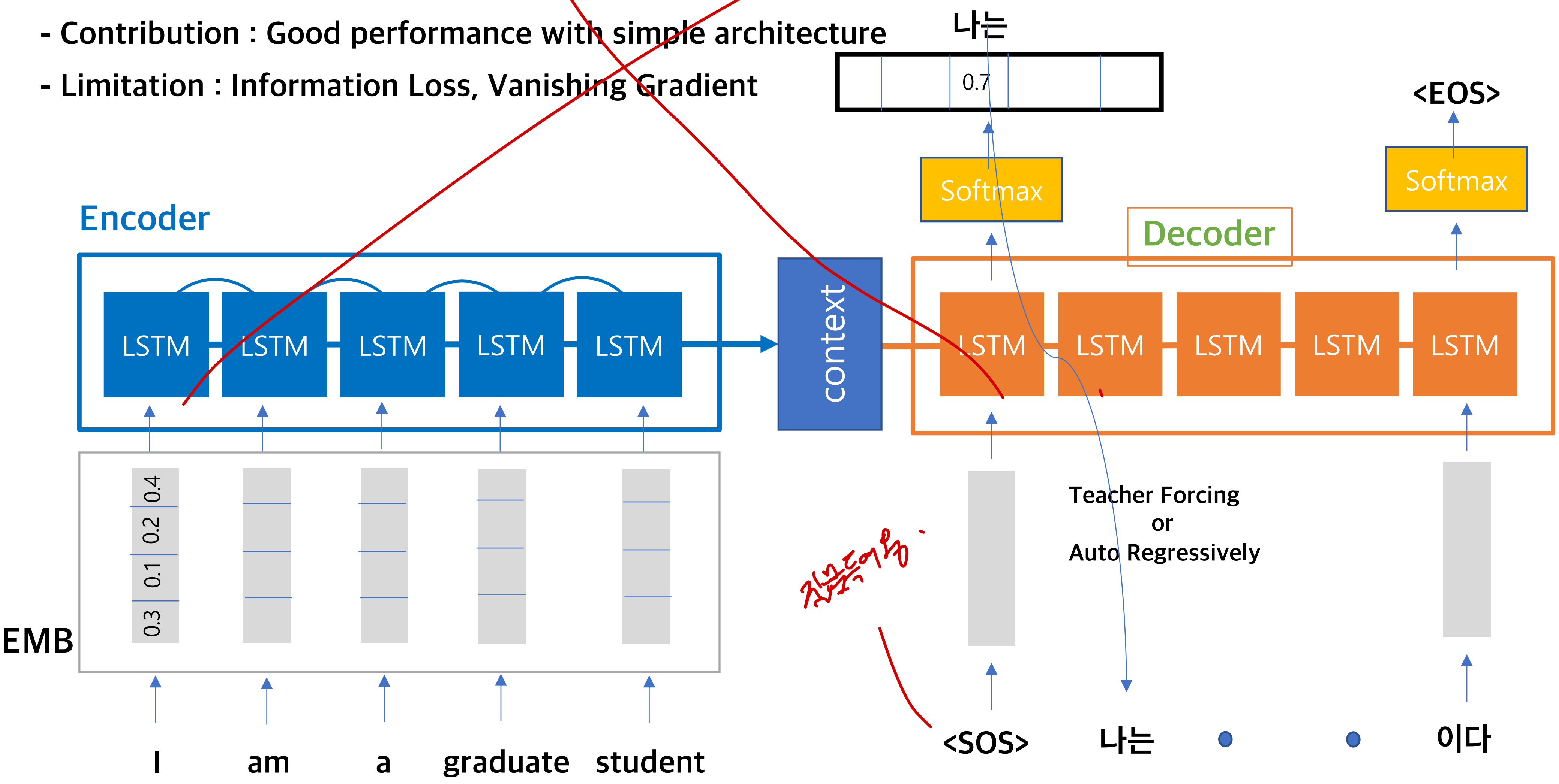
<Prior Work for long documents>

ltr : Left to Right

2. Presentation Streamline

❖ Encoder-Decoder

- Encode sequence and transforms to some sequence. (e.g., Machine translation, Text Summarization, Speech to Text)
- Loss Function : Cross-Entropy
- Contribution : Good performance with simple architecture
- Limitation : Information Loss, Vanishing Gradient

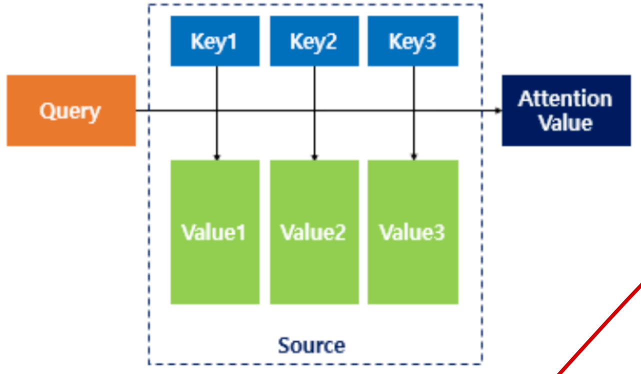


Sequence to Sequence Learning with Neural Networks

3. Background Knowledge : Attention Mechanism

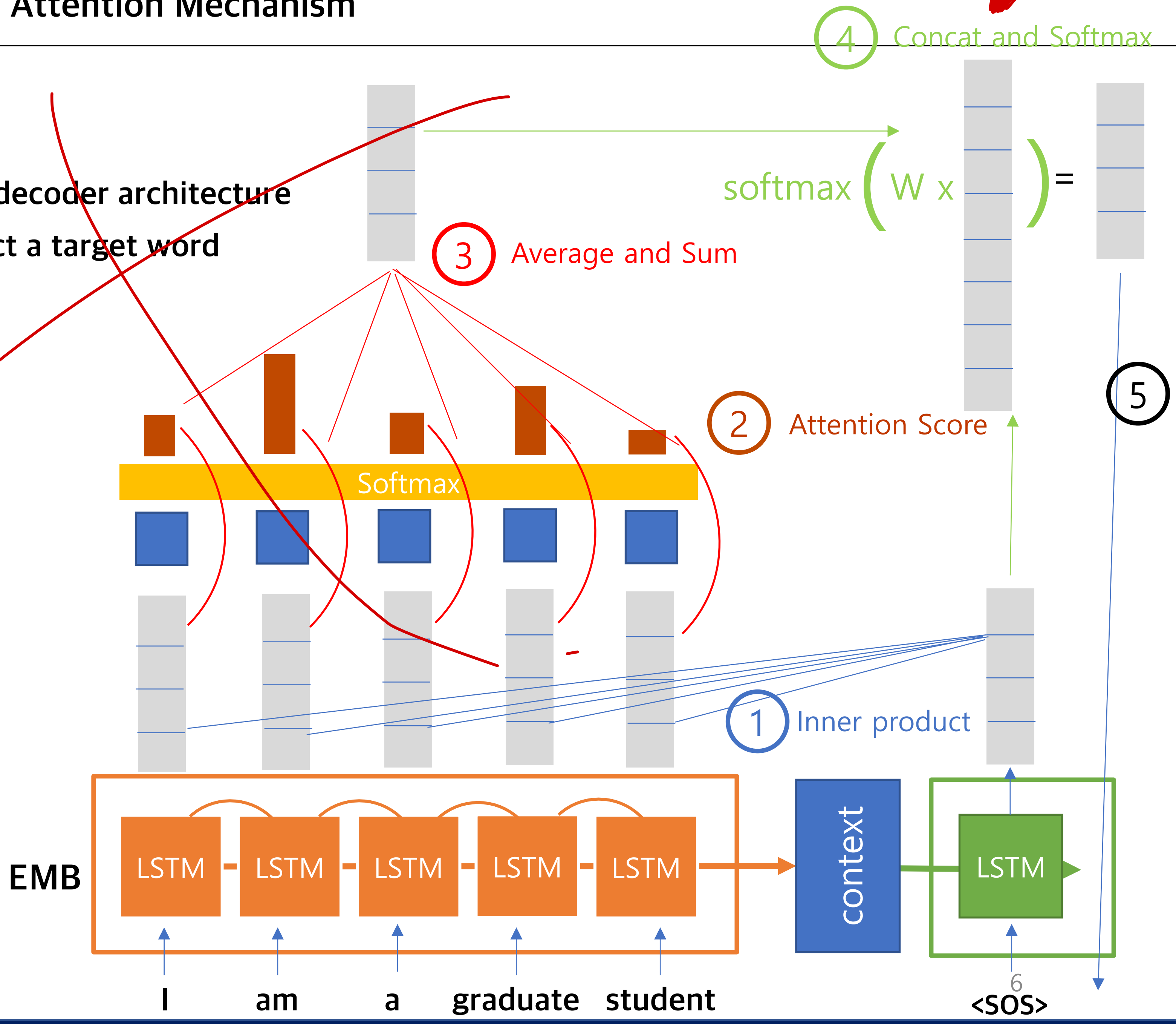
❖ Attention Mechanism

- Contribution
  - · Improving performance of encoder-decoder architecture
  - · Search relevant input parts to predict a target word



- Attention(Q,K,V) :
  - · Query - Decoder
  - · Key - Encoder
  - · Value - Encoder

- The basic idea of attention is that at every time step the decoder predicts an output word, the entire input sentence from the encoder is referenced once again.



Sequence to Sequence Learning with Neural Networks

 (방법2)

❖ Transformers : Encoder – Positional Encoding

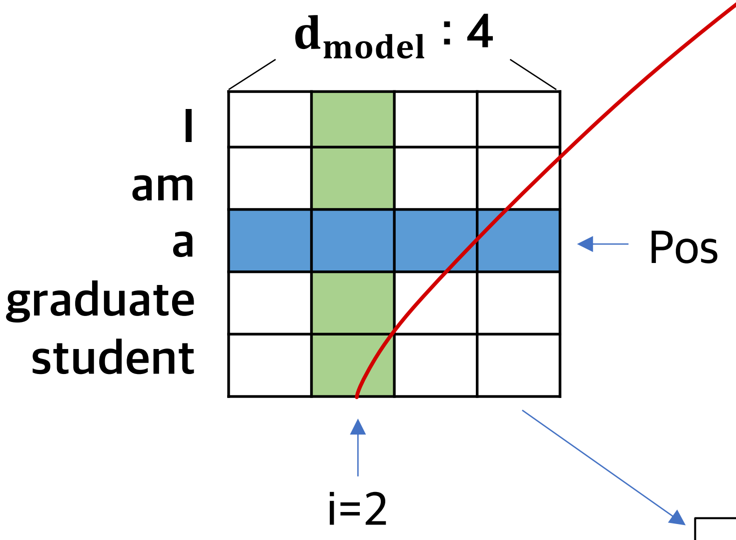
- Since model contains no recurrence, in order for the model to make use of the order of the sequence inject some information.

$$PE(pos, 2i) = sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right), \quad PE(pos, 2i+1) = cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

< even index feature >　　　< odds index feature >

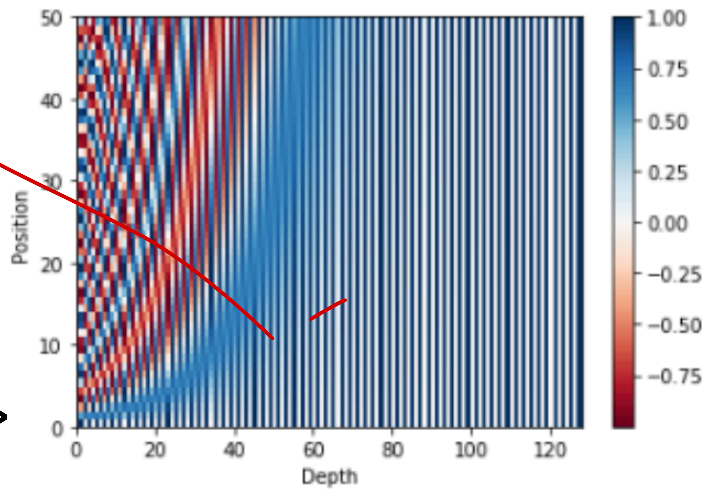
d_{model} : length of dimension
i : index of feature
Pos : position of words

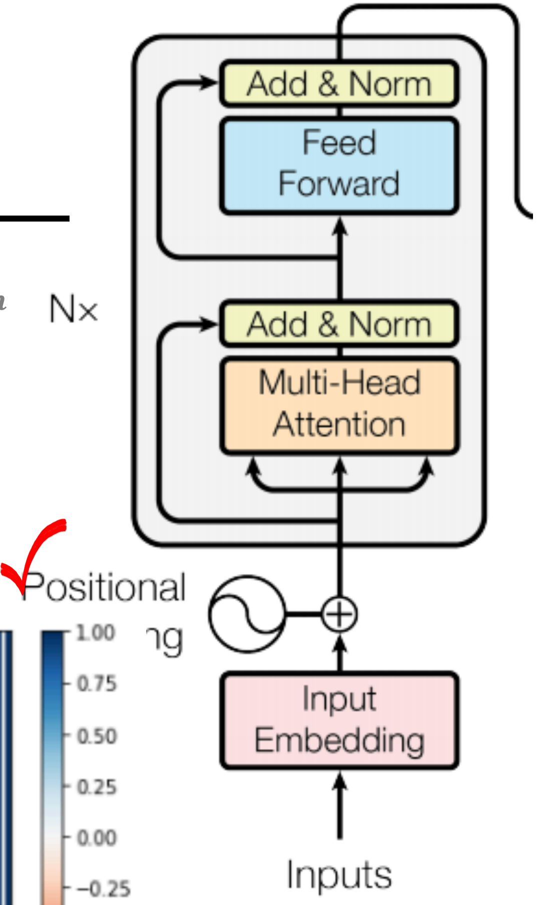
왜 겹치면 어떨까?



0.1	0.2	0.3	0.4
~~0.2~~	~~0.4~~	.	.
…	…	…	…
…	…	…	…
…	…	…	…

<Positional Embed>

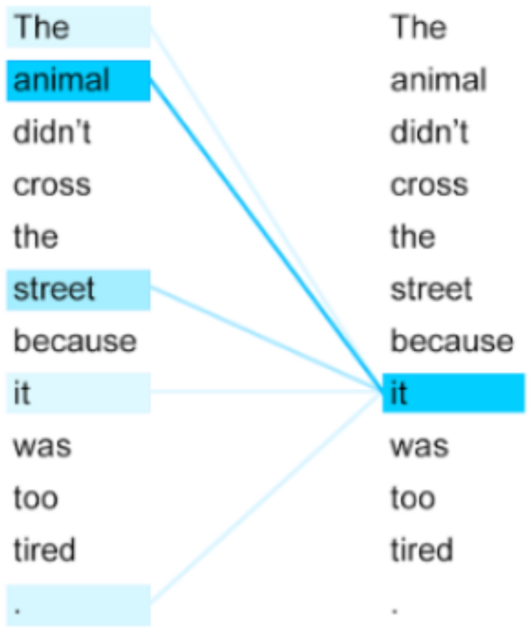


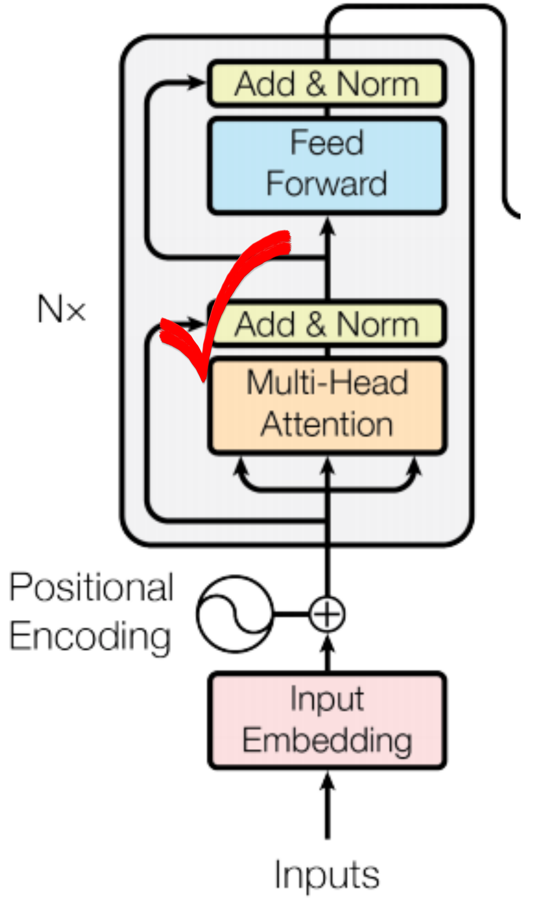


\+ Ready to plug in to the model

❖ Transformers : Encoder – Multi-Head Attention

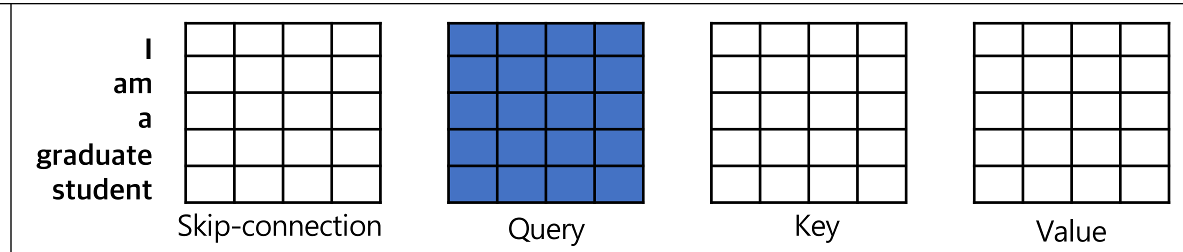
- Advantage : In Greek and Roman mythology, there is a multi-headed monster Hydra or Ceroberus. The characteristic of these monsters is that they have multiple heads, so you can see a lot from the viewpoint. There won't be much to be missed from this perspective, so it would be very difficult to attack these monsters. The same goes for multi-head attention. It is to collect information from a different perspective by viewing and performing attention.



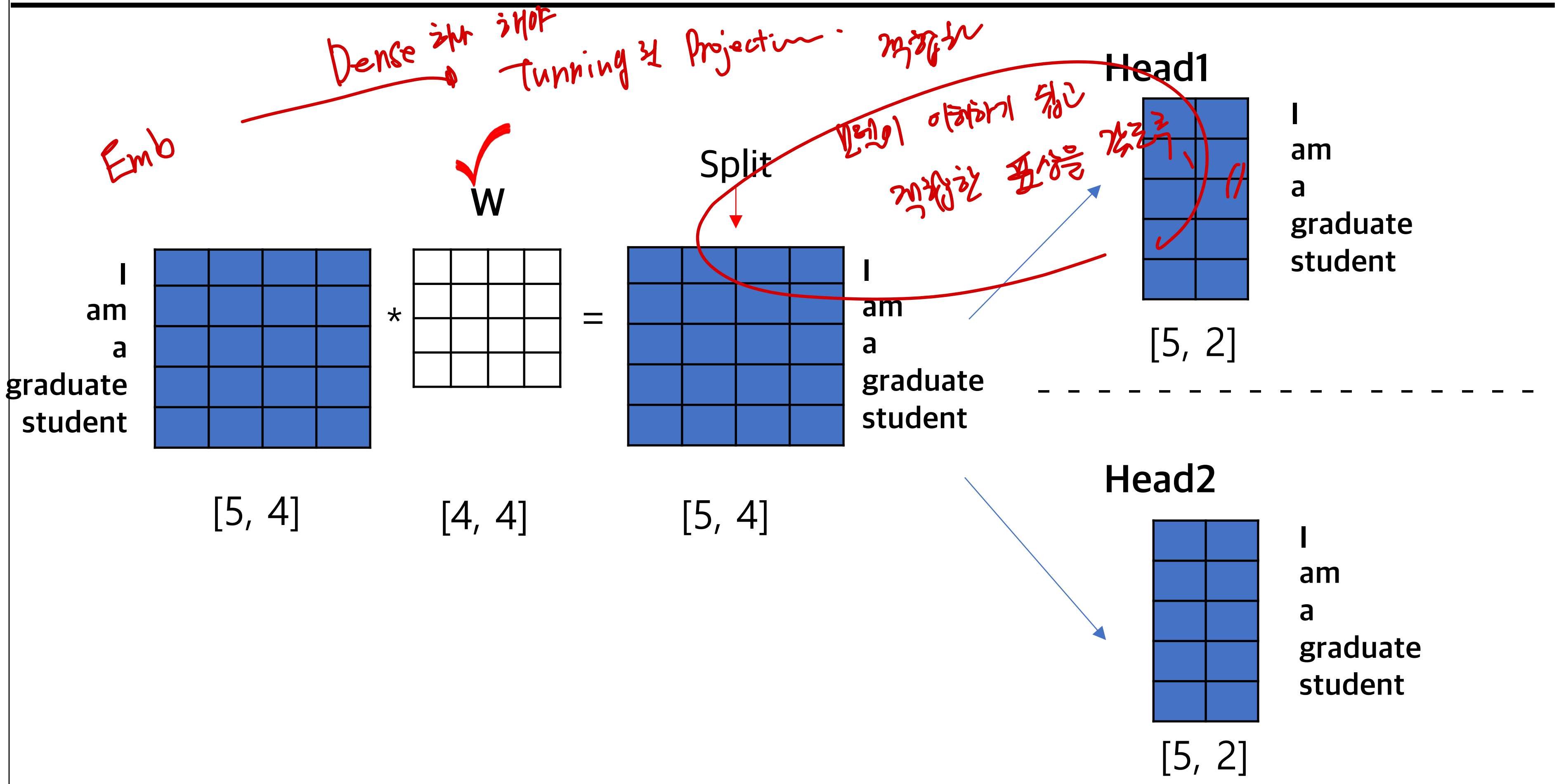


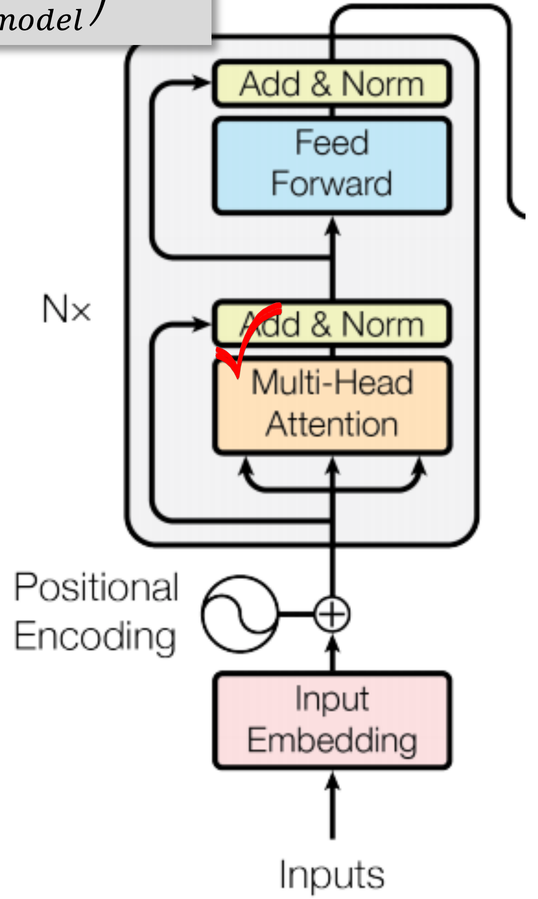
- Many of the attention heads attend to a distant dependency of the noun 'it' from 'animal', 'street', 'it'

3. Background Knowledge : Transformers(Attention Is All You Need)

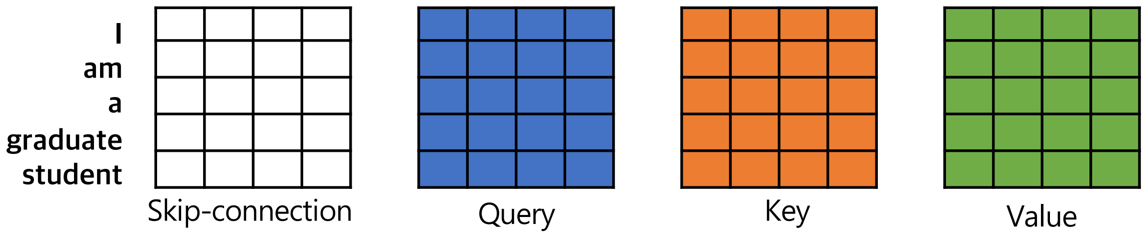


$$Attention(Q,K,V) = softmax\left(\frac{(Q*K^T)}{\sqrt{d_{model}}}\right)*V$$

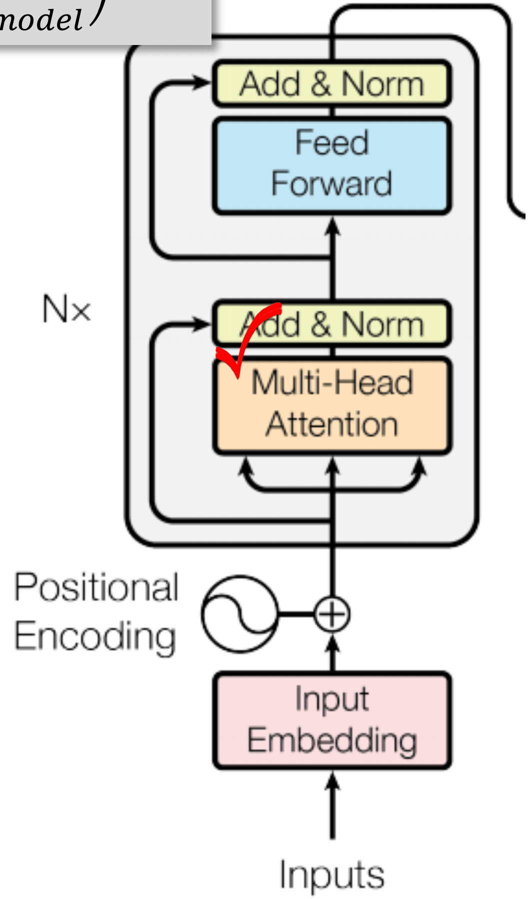


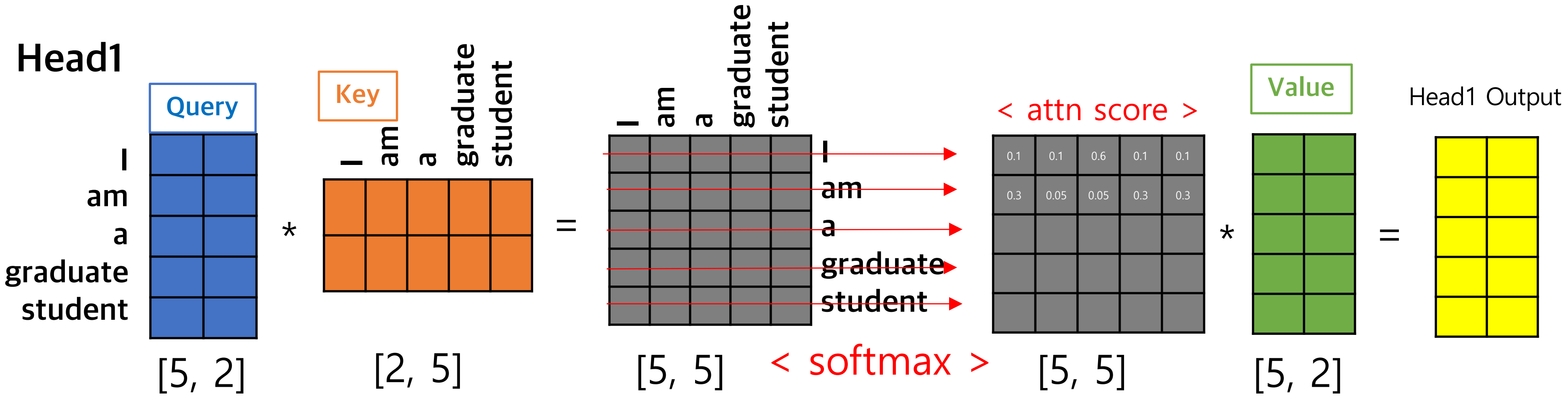


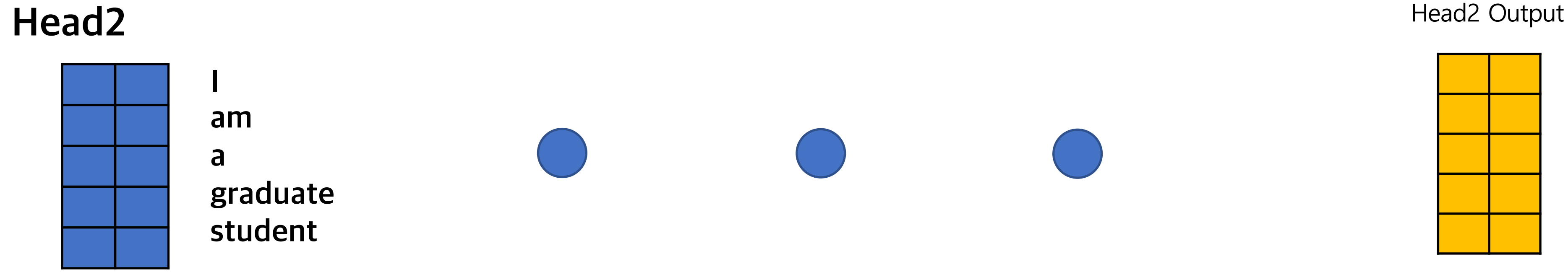
3. Background Knowledge : Transformers(Attention Is All You Need)



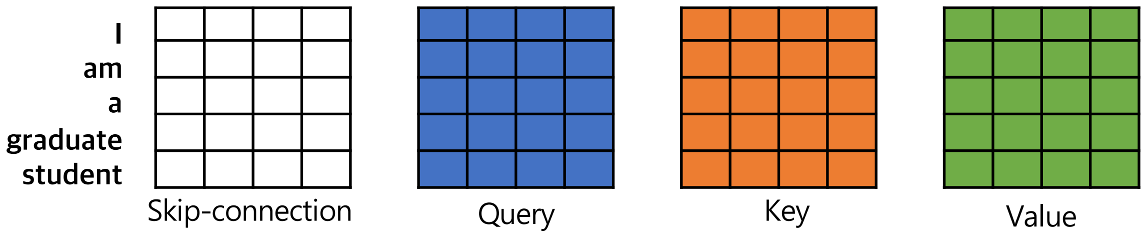
$$Attention(Q,K,V) = softmax\left(\frac{(Q * K^T)}{\sqrt{d_{model}}}\right) * V$$



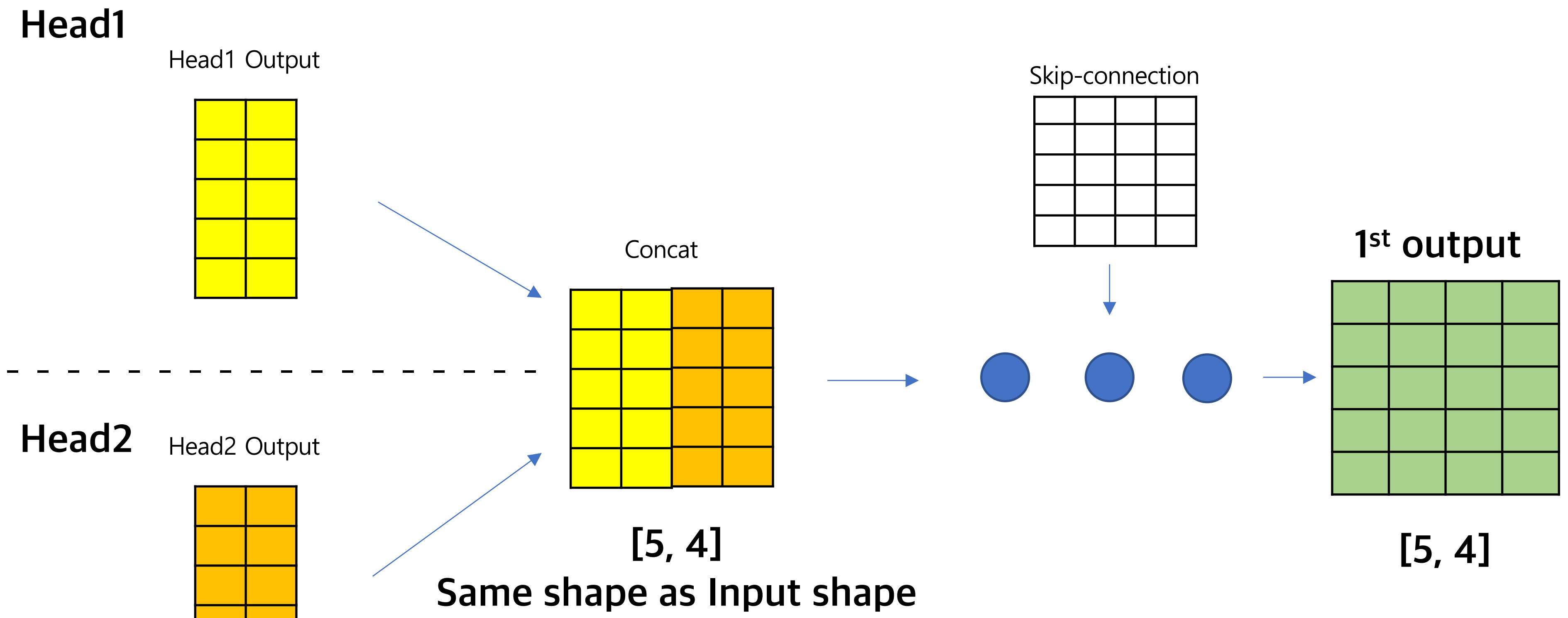


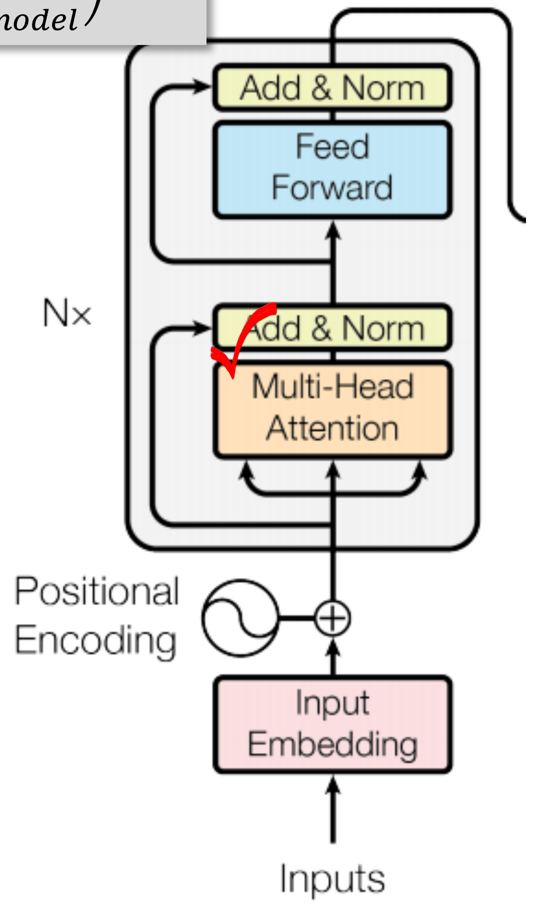


3. Background Knowledge : Transformers(Attention Is All You Need)



$$Attention(Q,K,V) = softmax\left(\frac{(Q * K^T)}{\sqrt{d_{model}}}\right) * V$$





3. Background Knowledge : Transformers(Attention Is All You Need)

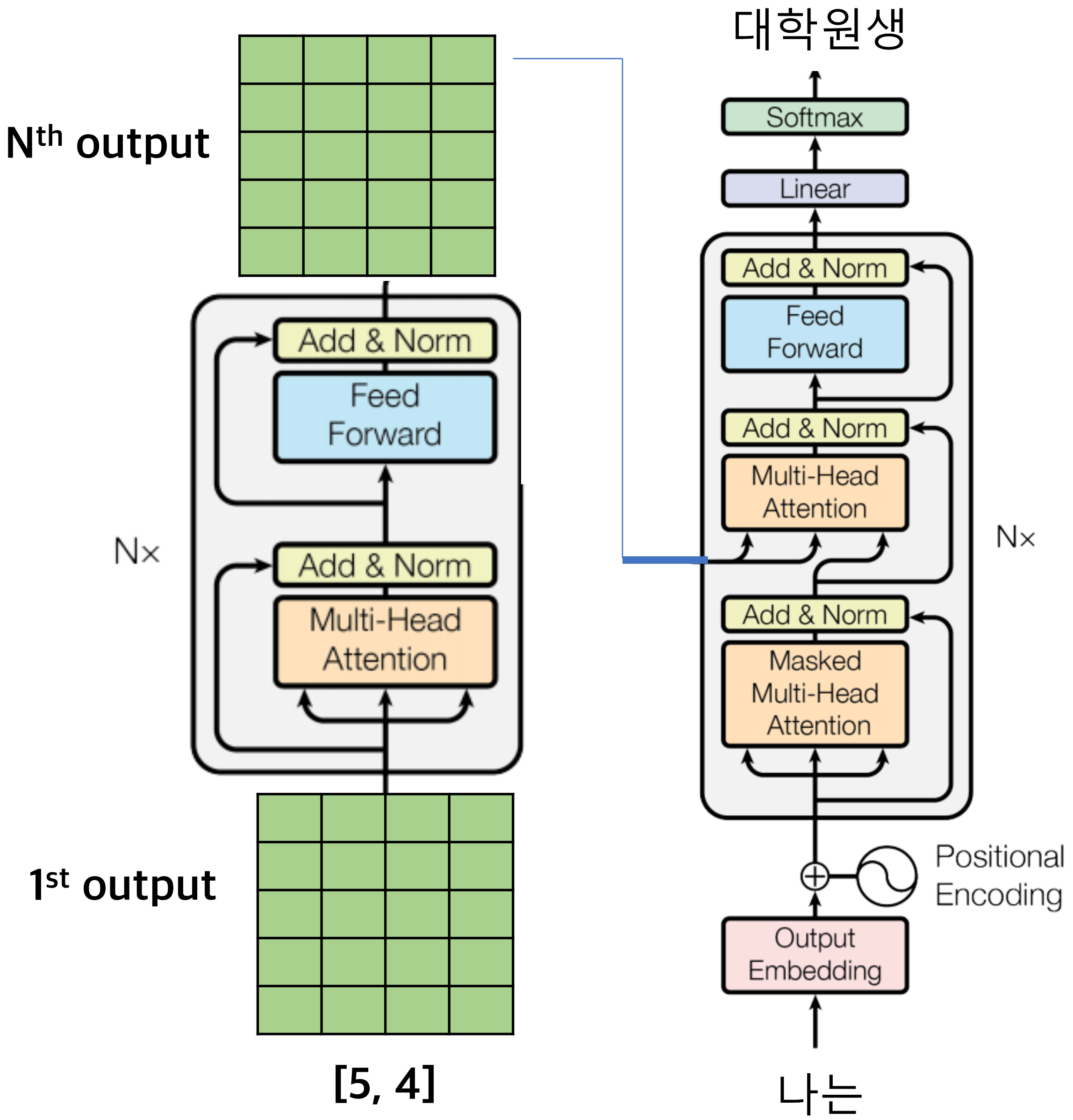
❖**Transformers**

- **Downstream Task for NLP**

  (Downstream : Use pretrained model for supervised-learning task)

- **Transformer's limitation**

  Uni-directional, predict next token from predicted tokens.

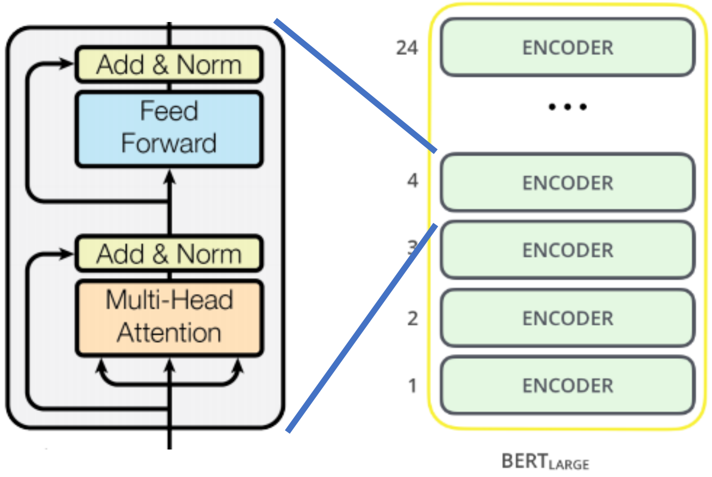


Attention Is All You Need

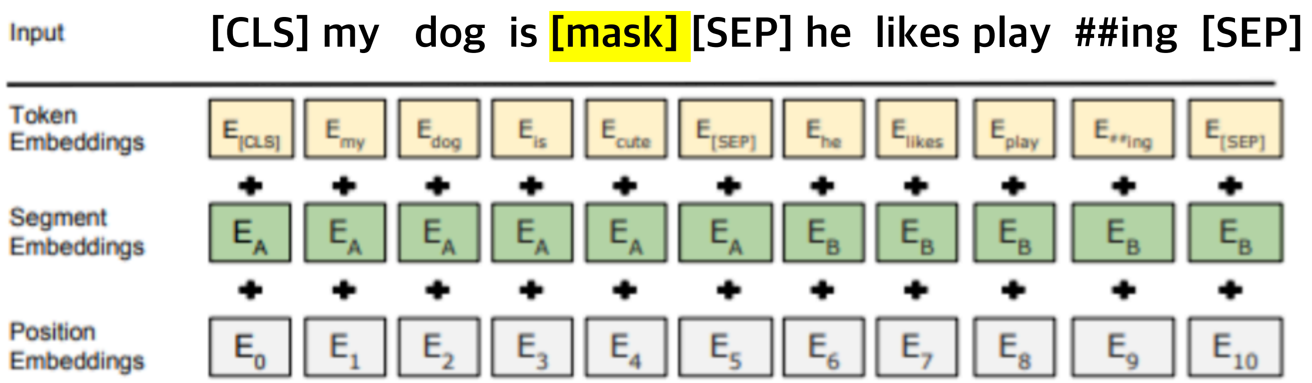
3. Background Knowledge : BERT

❖ BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding

- Bi-directional : Predict [mask] token referring from tokens
- Use only Encoder part of the Transformer.



- 15% of tokens are choose to be masked
  - · 80% of tokens become [MASK] token
  - · 10% of tokens are change to random token.
  - · 10% of tokens are not changed.



- Loss 1 : Cross-Entropy

  [CLS] - Predict correct next sentence

- Loss 2 : Cross-Entropy

  Predicted token and Ground Truth token.

4. Longformer

01

02

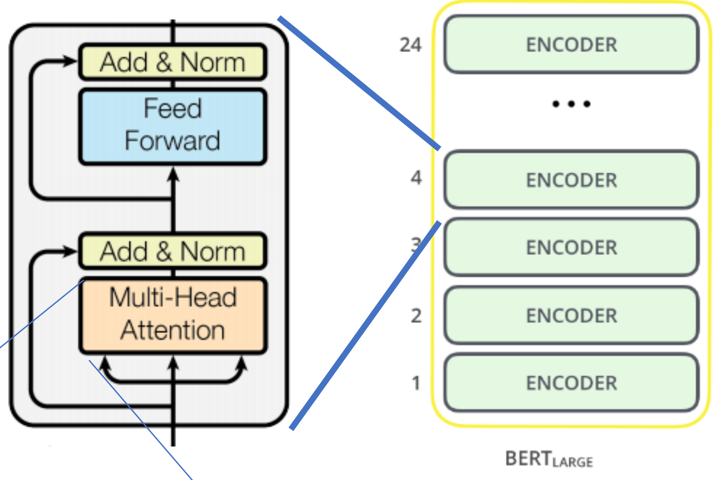
03

04

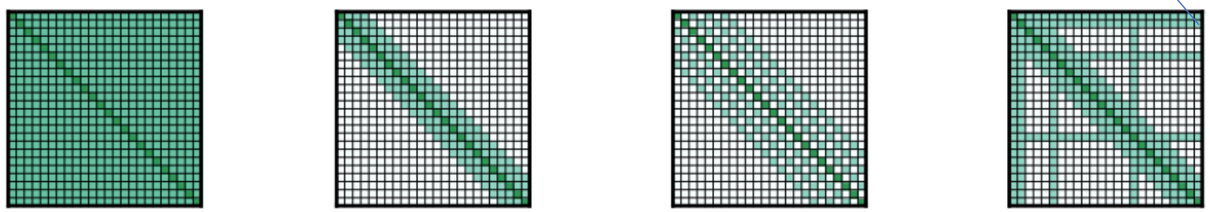
05

❖ LongFormer.

- Longformer gets high performance not only considering whole contextual information but also not depending on complex architecture.
- To address O(n^2) attention complexity, longformer proposes attention patterns which are 'Sliding Window', 'Dilated sliding window', and 'Global+sliding window'.
- Pretrain longformer from the RoBERTa which is BERT variant.



홀 컨텍스트를 참조 하든지 아닌지... 설명을 더 넣어야..

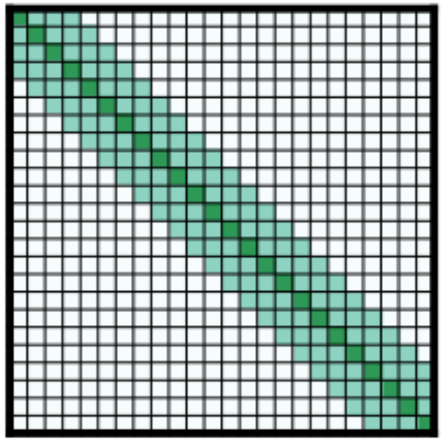


(a) Full n^2 attention (b) Sliding window attention (c) Dilated sliding window (d) Global+sliding window

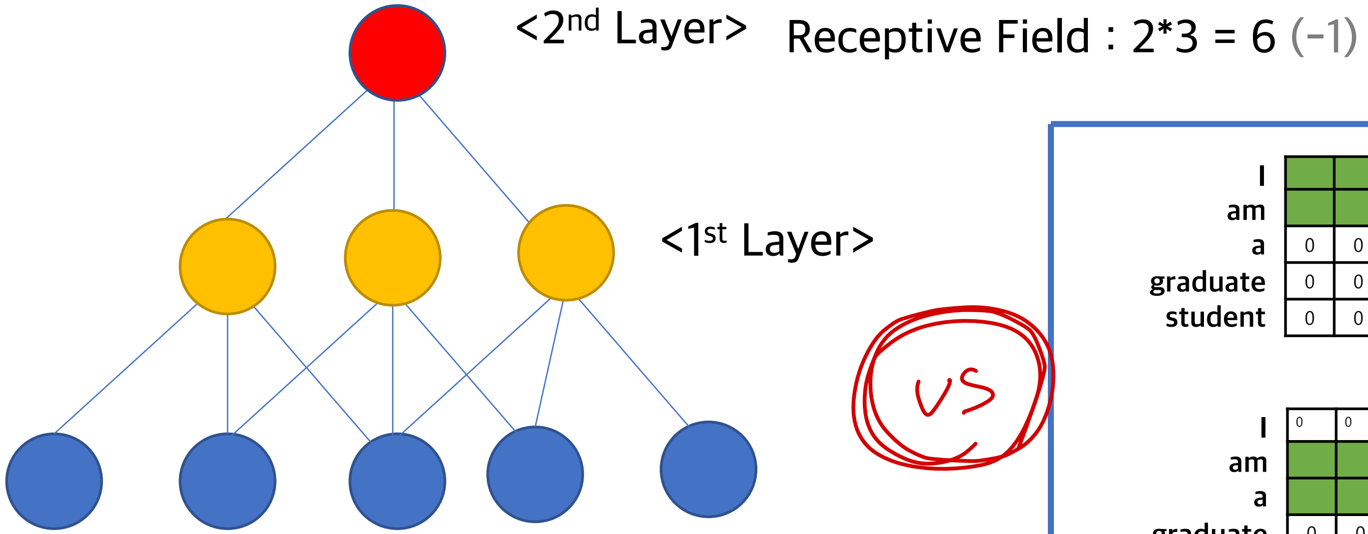
4. Longformer

❖ Sliding Window

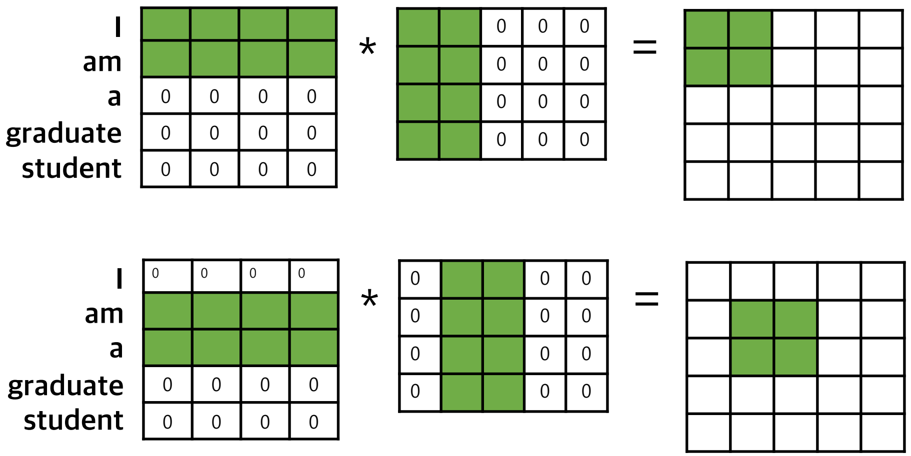
- Given a fixed window size w, each toke attends to 1/2*w tokens on each side(w was 512 on paper).
- The computation complexity : O(n * w)   n : number or sequence length
- Focus on local context.
- At the top of transformer layer, receptive field is l * w (-1)



(b) Sliding window attention



vs

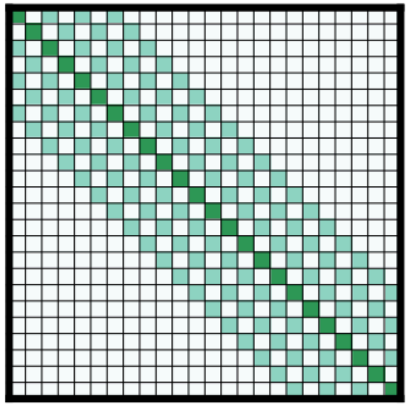


4. Longformer

❖ Dialated sliding window

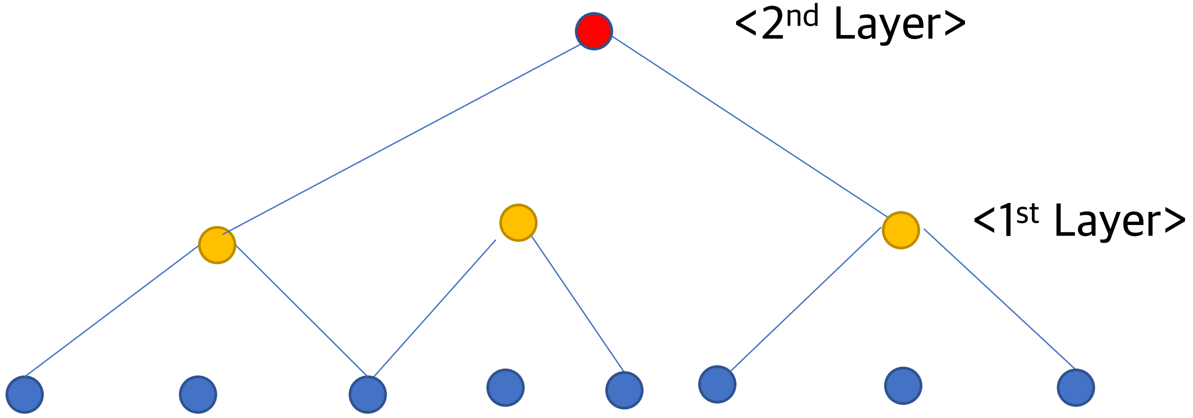
- To further increase the receptive field without increasing computation the sliding window can be 'dilated'
- At the top of transformer layer, receptive field is l*d*w
- Sliding window and Dialated sliding window methods make sense since CNN collect edge information on lower layer and combines those feature to higher features.

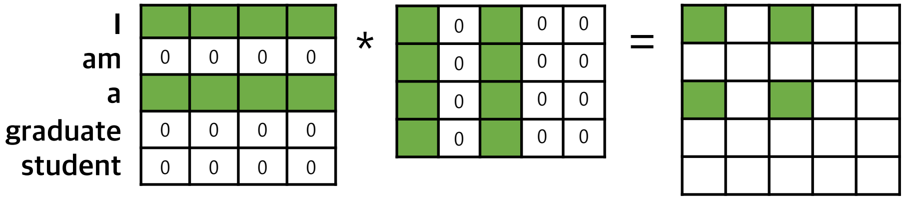
n : number or sequence length
d : dilated size



(c) Dilated sliding window

Receptive Field : 2*2*2 = 8

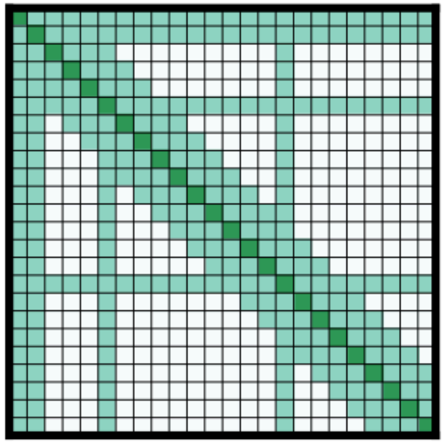




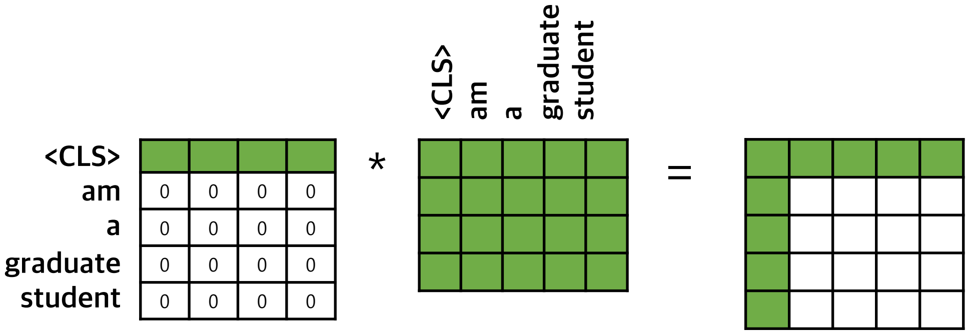
4. Longformer

❖ Global+sliding window

- BERT style models has special tokens such as [CLS], [SEP] for QA task, classification task.
- Those tokens need to consider whole tokens to solve specific problem which means those tokens need to be computed.



(d) Global+sliding window



4. Longformer

❖ Attention Pattern.

- Increasing the window size from the bottom to the top layer leads to the best performance
  - · Low Layer : Focus on Local context.
  - · Top Layer : Focus on whole context.
- Trade off between efficiency and performance

- On paper
  - · Low Layer : No dilation is applied.
  - · Top Layer : Dilation is applied only two heads.

- 5 phases
  - · Before training longer context, local context should be trained first.
  - · 2,048 long token sequence is trained on first phase with small window size
  - · Double the sequence length and window size.
  - · Train until 23,040 long token sequence.

BPC is average cross-entropy

$$bpc(string) = \frac{1}{T}\sum_{t=1}^{T} H(P_t, \hat{P}_t) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{c=1}^{n} P_t(c)\log_2 \hat{P}_t(c),$$
$$= -\frac{1}{T}\sum_{t=1}^{T} \log_2 \hat{P}_t(x_t).$$

T is the length of your input string
c is possible character

Model	Dev BPC
Decreasing w (from 512 to 32)	1.24
Fixed w (= 230)	1.23
Increasing w (from 32 to 512)	**1.21**
No Dilation	1.21
Dilation on 2 heads	**1.20**

5. Results

어떤건지 찾아보기.

- State Of The Art on both text8 and enwik8

Model	#Param	Dev	Test
Dataset `text8`			
T12 (Al-Rfou et al., 2018)	44M	-	1.18
Adaptive (Sukhbaatar et al., 2019)	38M	1.05	1.11
BP-Transformer (Ye et al., 2019)	39M	-	1.11
Our Longformer	41M	1.04	**1.10**
Dataset `enwik8`			
T12 (Al-Rfou et al., 2018)	44M	-	1.11
Transformer-XL (Dai et al., 2019)	41M	-	1.06
Reformer (Kitaev et al., 2020)	-	-	1.05
Adaptive (Sukhbaatar et al., 2019)	39M	1.04	1.02
BP-Transformer (Ye et al., 2019)	38M	-	1.02
Our Longformer	41M	1.02	**1.00**

Table 2: *Small* model BPC on `text8` & `enwik8`

- Similar performance SOTA record but has much less Parameters.

Model	#Param	Test BPC
Transformer-XL (18 layers)	88M	1.03
Sparse (Child et al., 2019)	≈100M	0.99
Transformer-XL (24 layers)	277M	0.99
Adaptive (Sukhbaatar et al., 2019)	209M	0.98
Compressive (Rae et al., 2020)	277M	0.97
Routing (Roy et al., 2020)	≈223M	0.99
Our Longformer	102M	0.99

Table 3: Performance of *large* models on `enwik8`