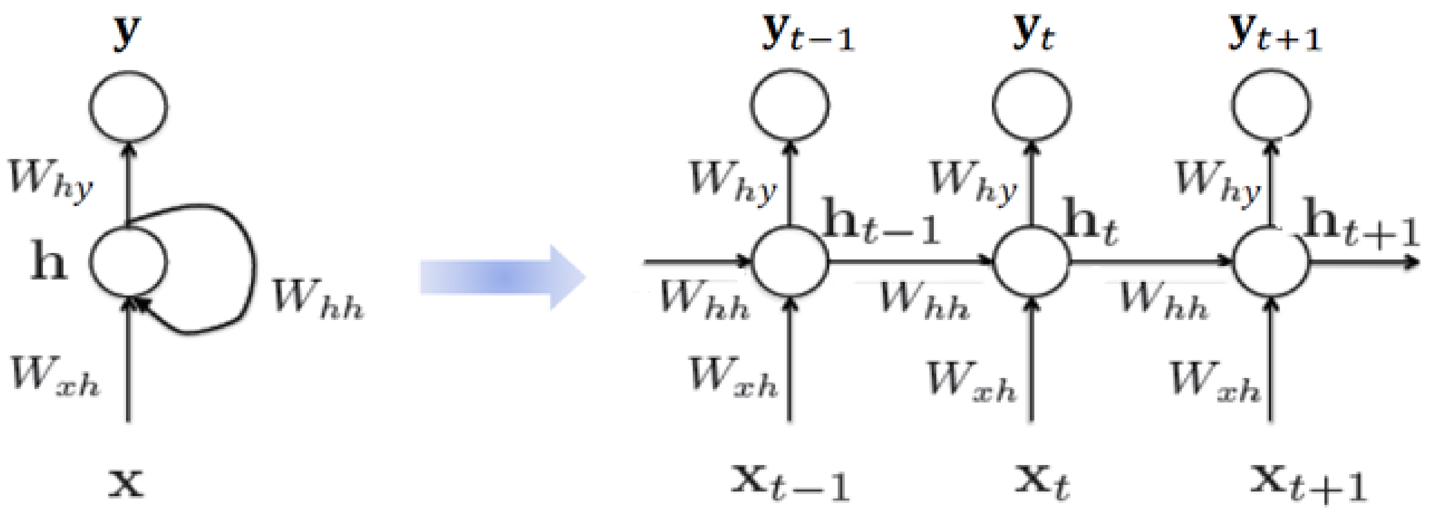
신입생 Deep Learning 기초 교육

5회: Transformer

Multimodal Language Cognition Lab,
Kyungpook National University

2023.02.13

Recurrent Neural Networks(RNN)



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

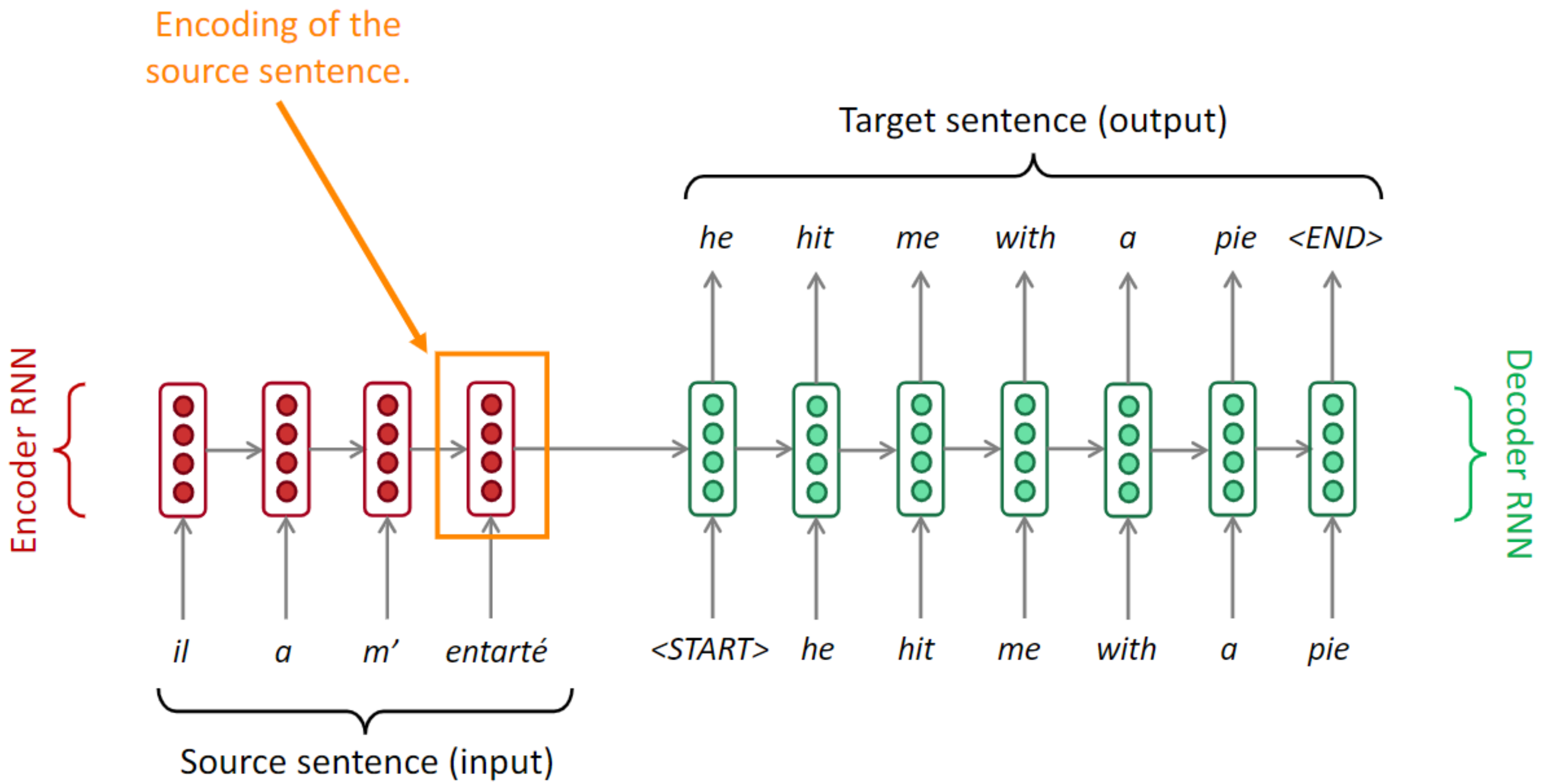
$$\downarrow$$

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

$$y_t = W_{hy}h_t$$

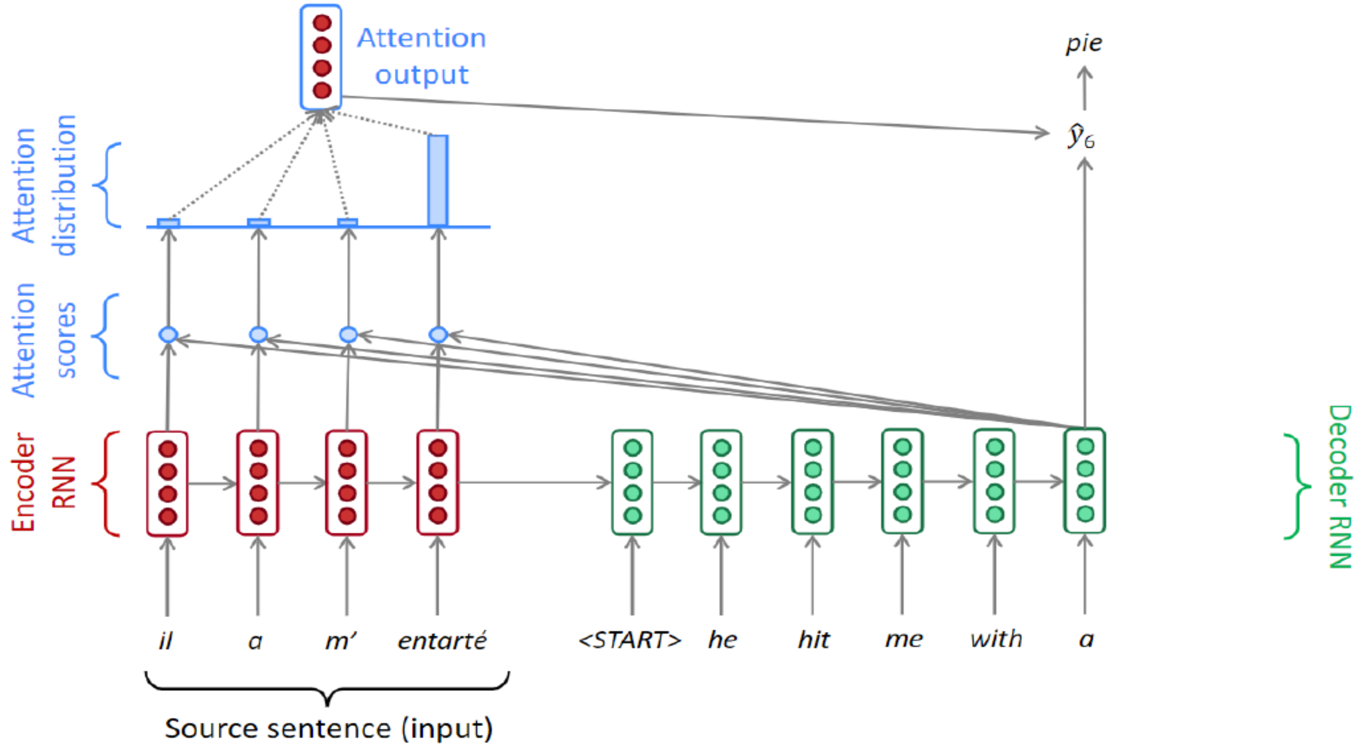
- W_{xh} = Weights connecting the input layer and hidden layer
- W_{hh} = Weights connecting the hidden layer and hidden layer
- W_{hy} = Weights connecting the hidden layer and output layer
- **Weight sharing**
  - The amount of parameters in the model is reduced
  - Independent of the length of the feature vector T

Sequence-to-sequence



Problems with Seq2Seq ?

Sequence-to-sequence with attention

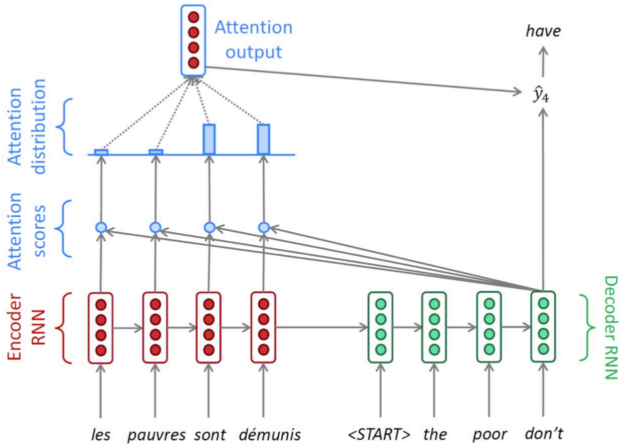


- **Attention** provides a solution to the bottleneck problem
- Core idea

  On each step of the decoder, use direct connection to the encoder to focus on a particular part of the source sequence

Background

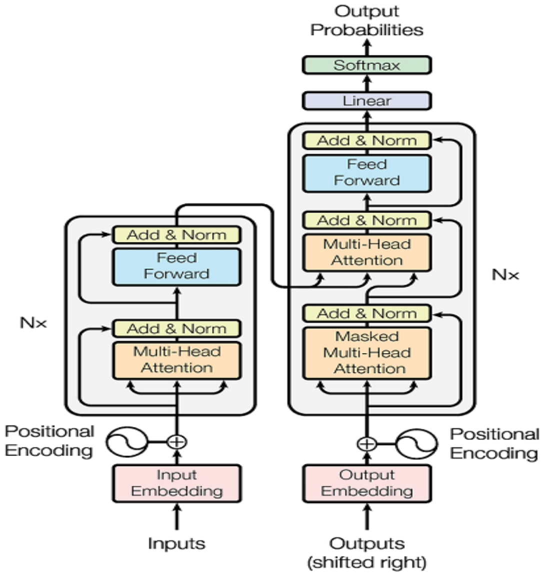
Sequence to Sequence with attention



RNN problems

- Long term dependency
- Fixed context vector
- Vanishing gradient
- Sequential processing

Transformer



Only Attention

Background

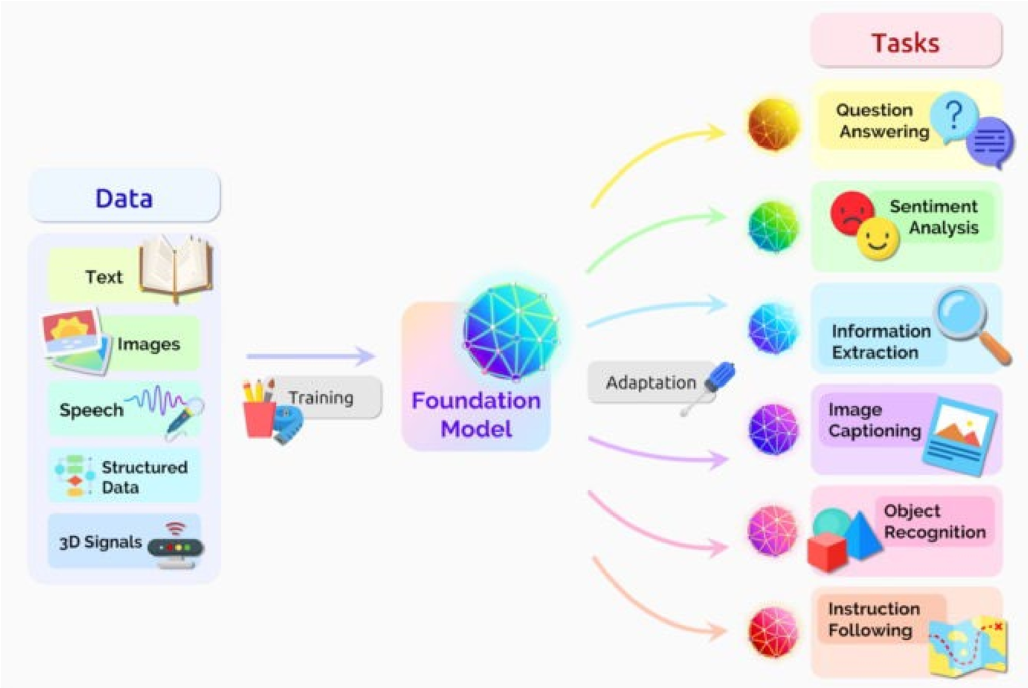


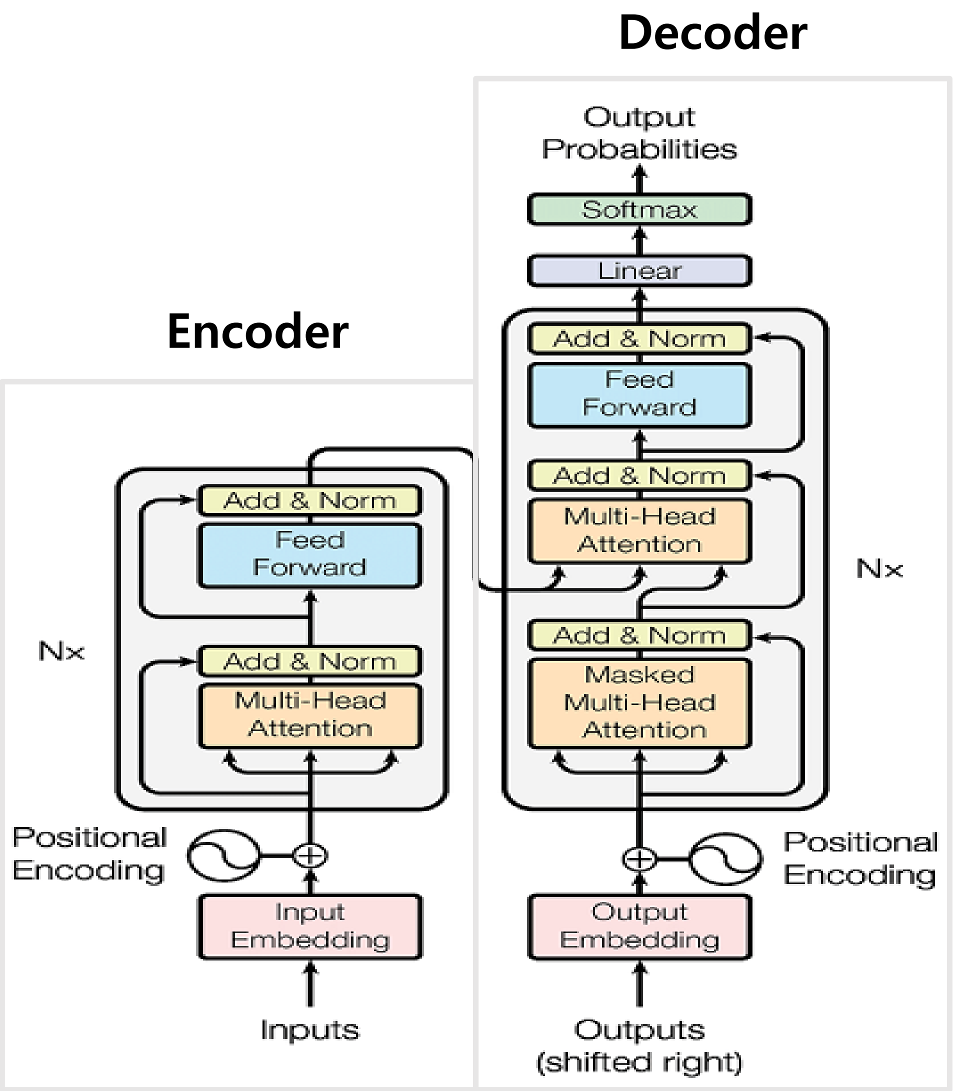
Image Classification

Rank	Model	Percentage correct	PARAMS	Extra Training Data	Paper	Code	Result	Year	Tags
1	ViT-H/14	99.5	632M	✓	An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale			2020	Transformer
2	ViT-L/16	99.42	307M	✓	An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale			2020	Transformer
3	CaiT-M-36 U 224	99.4		✓	Going deeper with Image Transformers			2021	Transformer
4	CvT-W24	99.39		✓	CvT: Introducing Convolutions to Vision Transformers			2021	Transformer

Sentiment Analysis

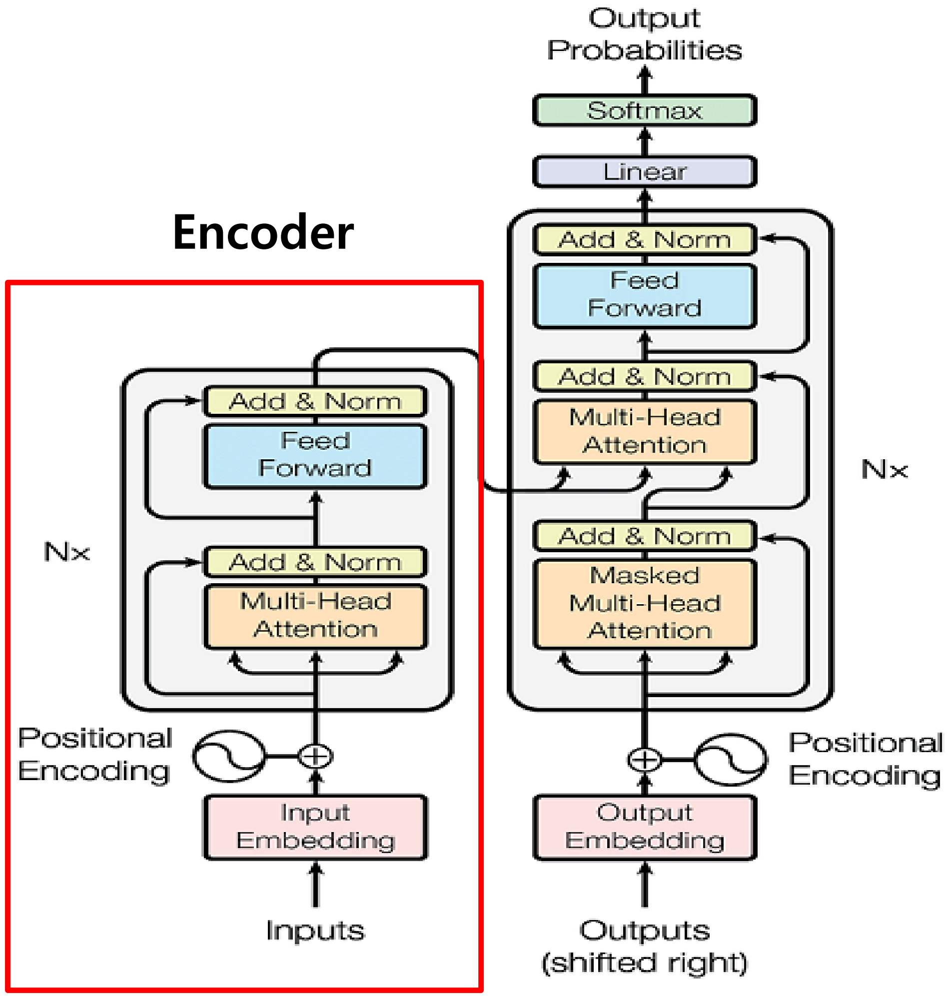
Rank	Model	Accuracy	Paper	Code	Result	Year	Tags
1	SMART-RoBERTa Large	97.5	SMART: Robust and Efficient Fine-Tuning for Pre-trained Natural Language Models through Principled Regularized Optimization			2019	Transformer
2	T5-3B	97.4	Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer			2019	Transformer

Transformer



- The encoder maps an input sequence of symbol representations $(x_1, \ldots, x_n)$ to a sequence of continuous representations $z = (z_1, \ldots, z_n)$
- Given z, the decoder then generates an output sequence $(y_1, \ldots, y_m)$ of symbols one element at a time

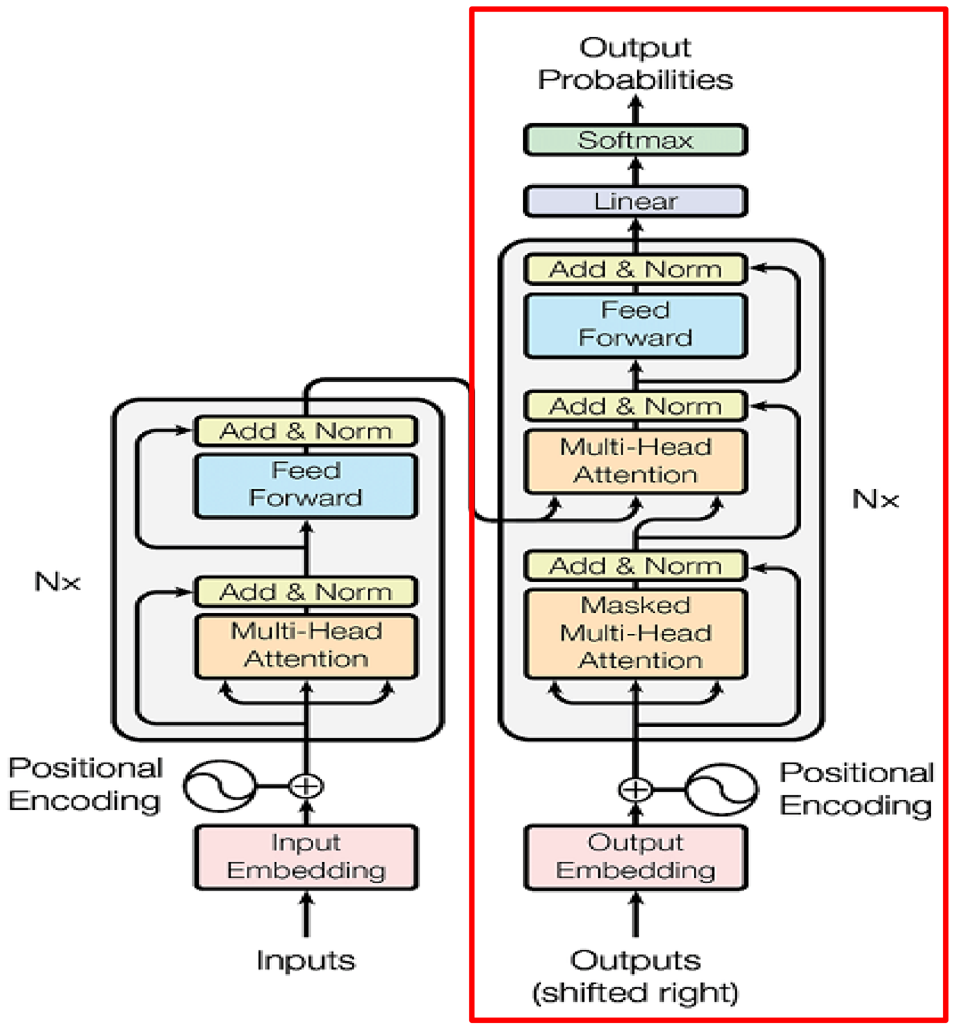
Transformer Encoder



- The encoder is composed of a stack of N = 6 identical layers
- Each layer has two sub-layer, **multi-head attention** and feed-forward network
- The output of each sub-layer is $\boldsymbol{LayerNorm}(\boldsymbol{x} + \boldsymbol{Sublayer}(\boldsymbol{x}))$
- All sub-layers in the model, as well as the embedding layers, produce outputs of dimension $\boldsymbol{d_{model}} = \mathbf{512}$

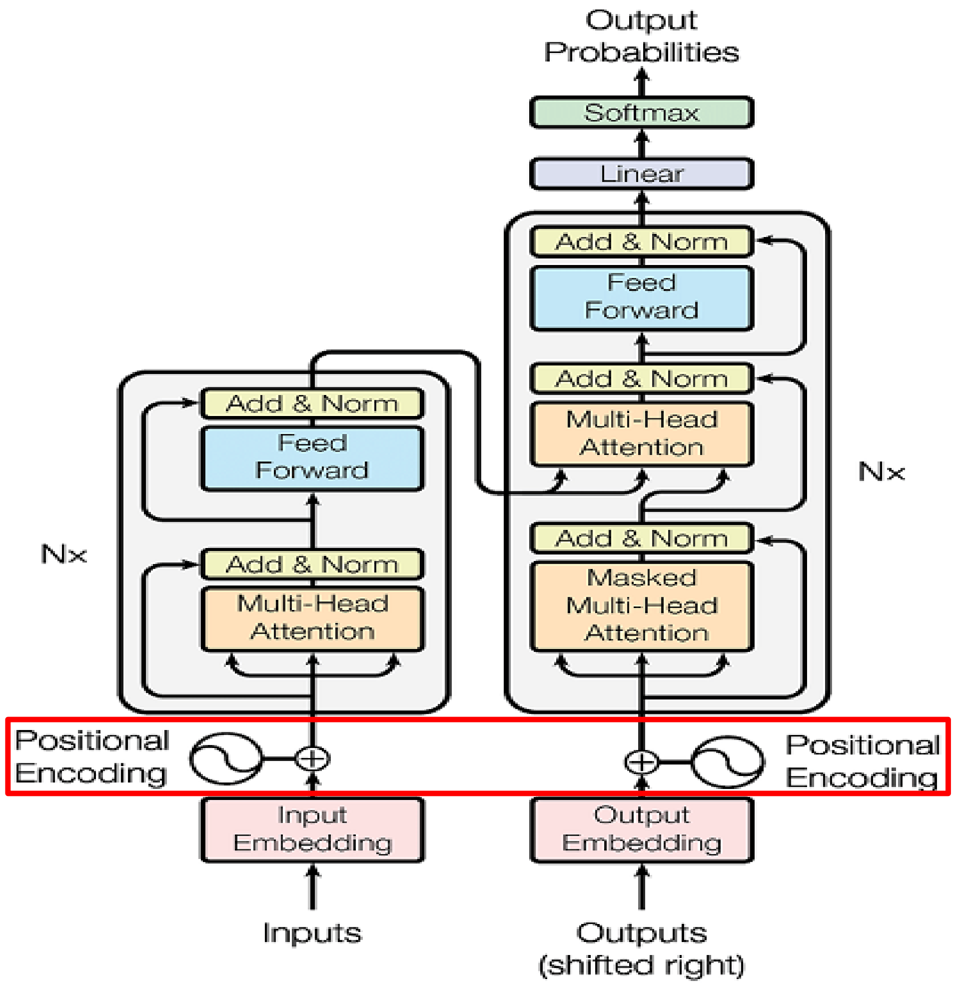
Transformer Encoder

Decoder



- The encoder is composed of a stack of N = 6 identical layers
- Each layer has two sub-layer, **masked multi-head attention, multi-head attention** and feed-forward network
- The output of each sub-layer is $\boldsymbol{LayerNorm(x + Sublayer(x))}$
- All sub-layers in the model, as well as the embedding layers, produce outputs of dimension $\boldsymbol{d_{model} = 512}$

Transformer Positional Encoding



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

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

- Transformer receives all sequences as input at once
- It does not take into account **the positional information** of words

Transformer Positional Encoding

pos = 1	This				
pos = 2	is				
pos = 3	song				
pos = 4	of				
pos = 5	BTS				
pos = 6	it				
pos = 7	is				
pos = 8	fantastic				

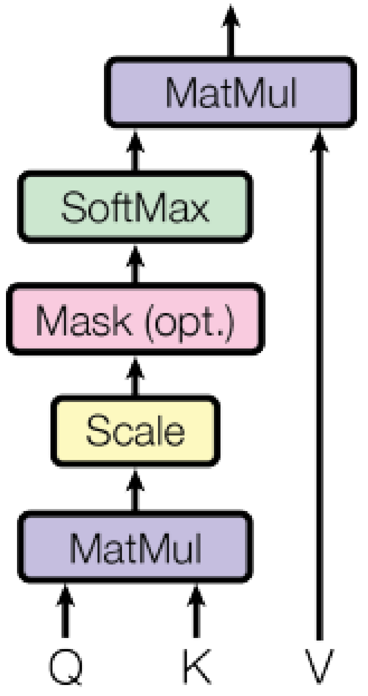
$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}})$$

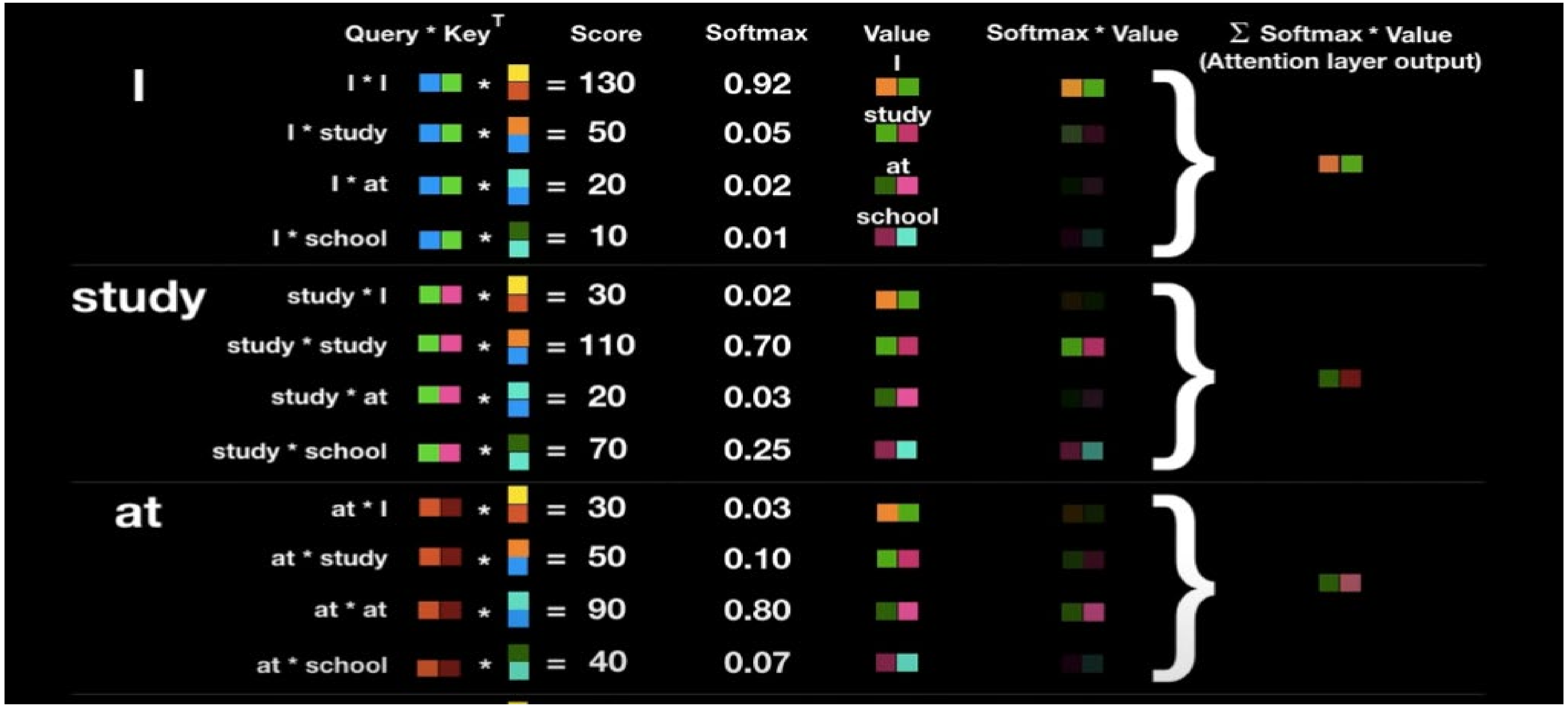
$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}})$$

- Transformer receives all sequences as input at once
- It does not take into account **the positional information** of words

Transformer Attention

Scaled Dot-Product Attention





Transformer Attention

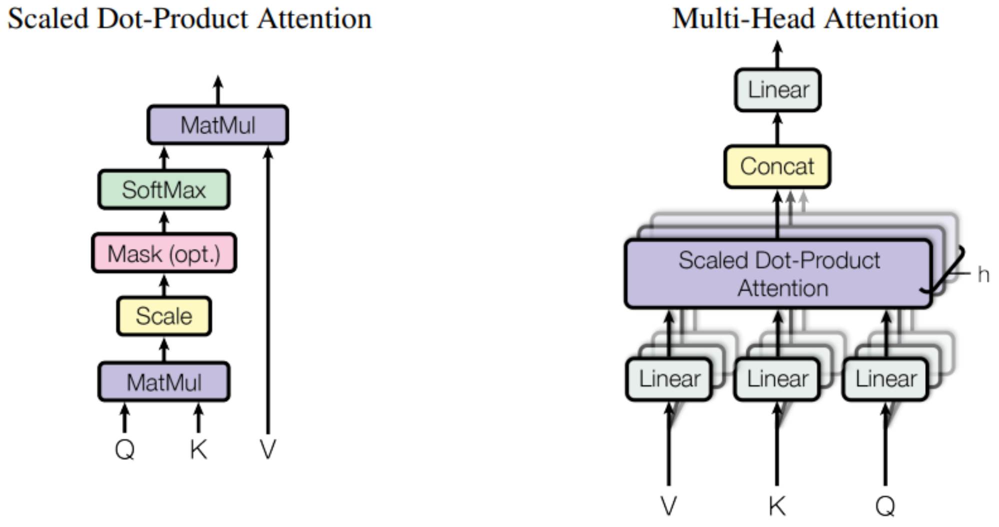
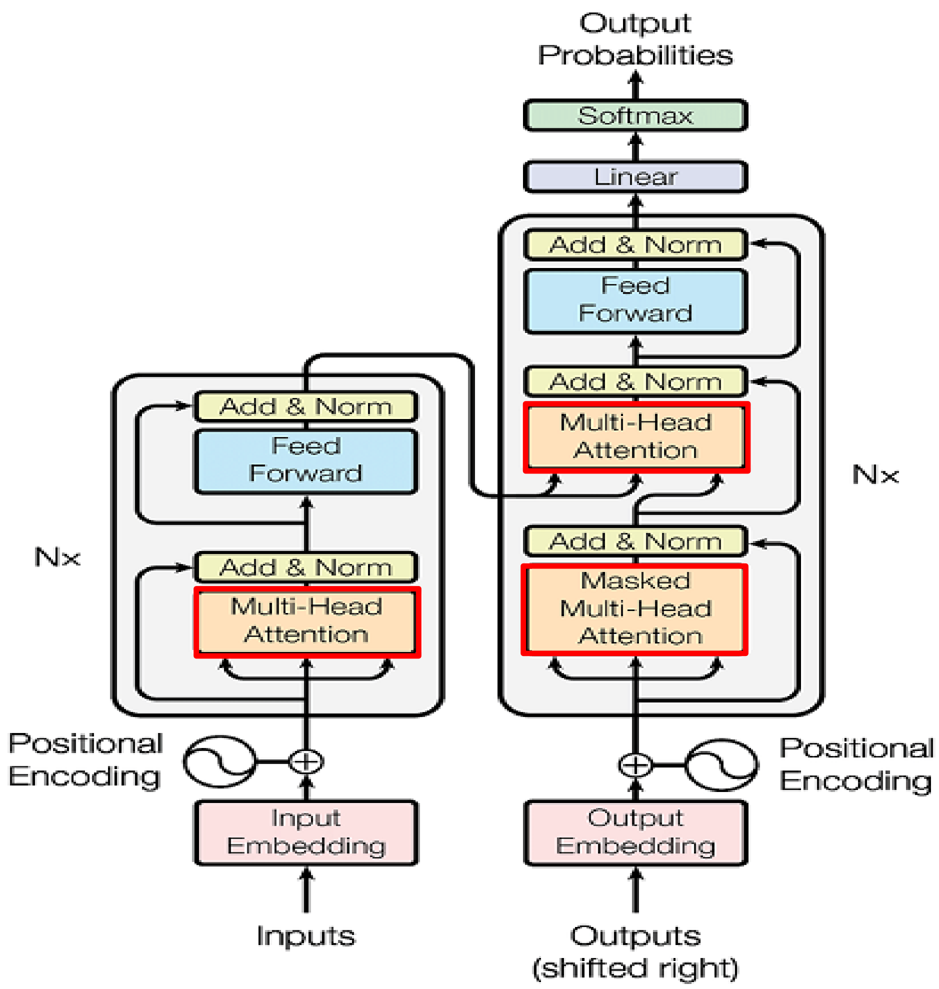


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

- Q=Current token, K =Target token, V=Target token
- $Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$
- $MultiHead(Q,K,V) = Concat(head_1, \dots, head_h)W^O$

  $where\ head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$
- In this work we employ $h = 8$ parallel attention layers, or heads
- For each of these we use $d_{\mathrm{model}} = 512$, $d_k = d_v = d_{model}/h = 64$

Transformer Attention

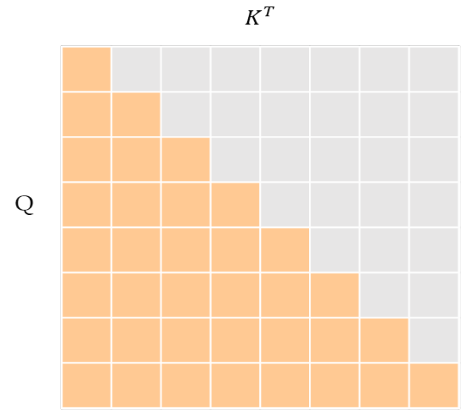


- **Encoder Multi-Head Attention ?**

  Q=K=V= encoder input sentence vector

- **Decoder Masked Multi-Head Attention**

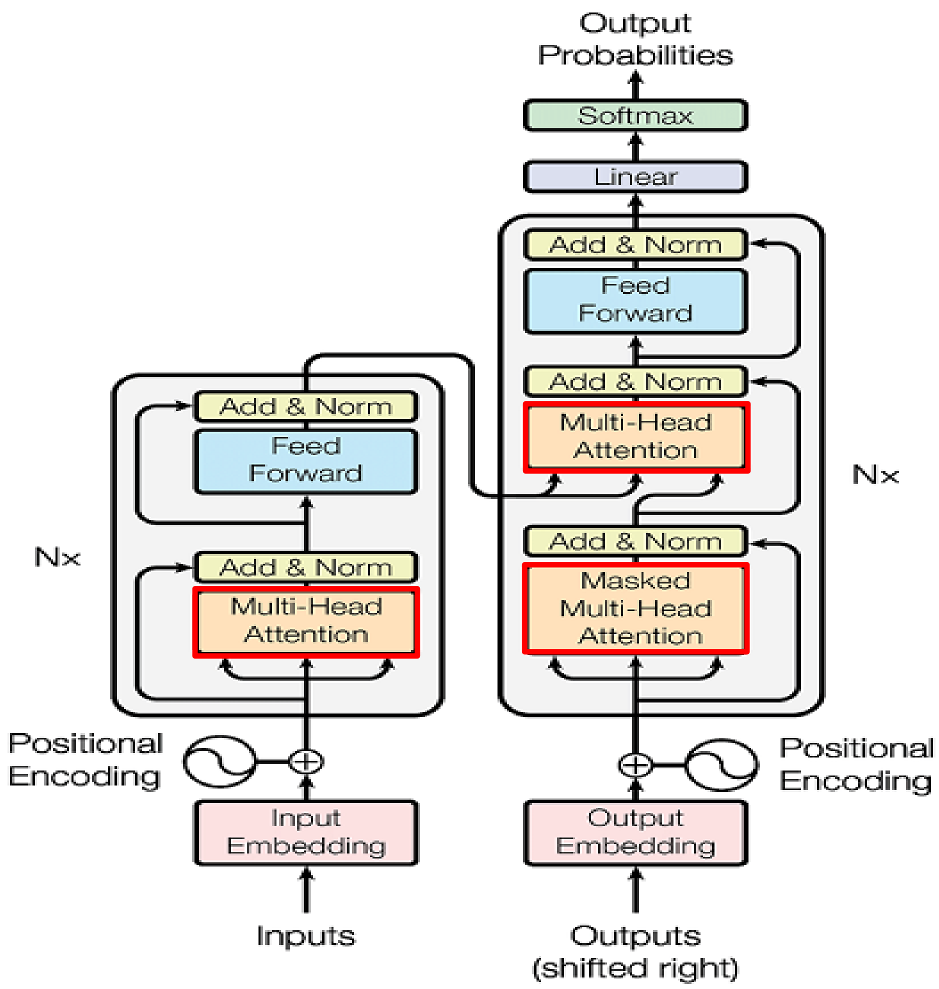
  Q=K=V=decoder input sentence vector



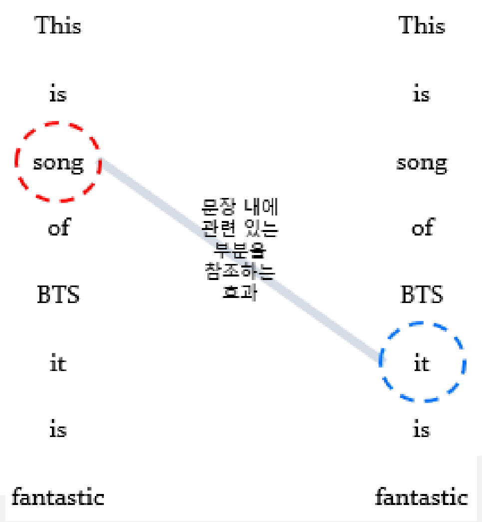
- **Decoder Multi-Head Attention**

  Q=Decoder vector , K=V=Encoder vector

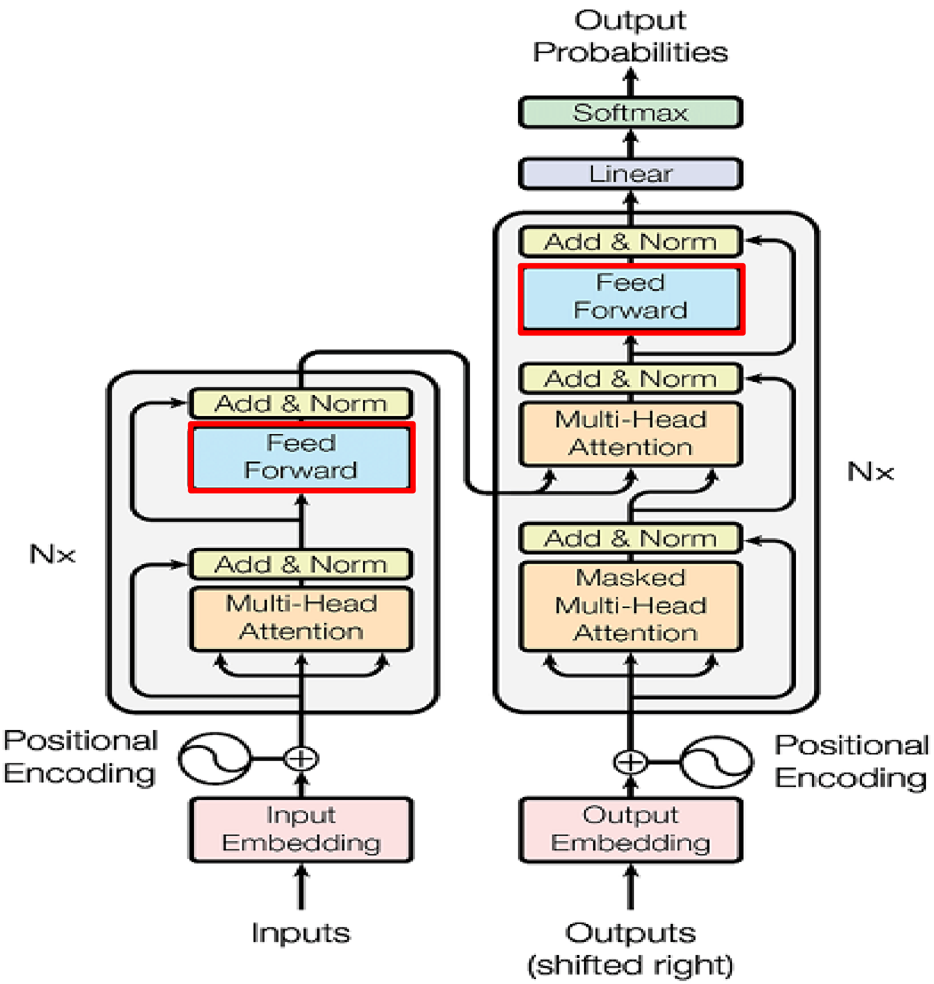
Transformer Attention



- **Multi-Head Attention**
  - On each of these projected versions of queries, keys and values we then perform the attention function in **parallel**
  - Multi-head attention allows the model to jointly attend to information from **different representation subspaces** at different positions
- **Self-Attention**

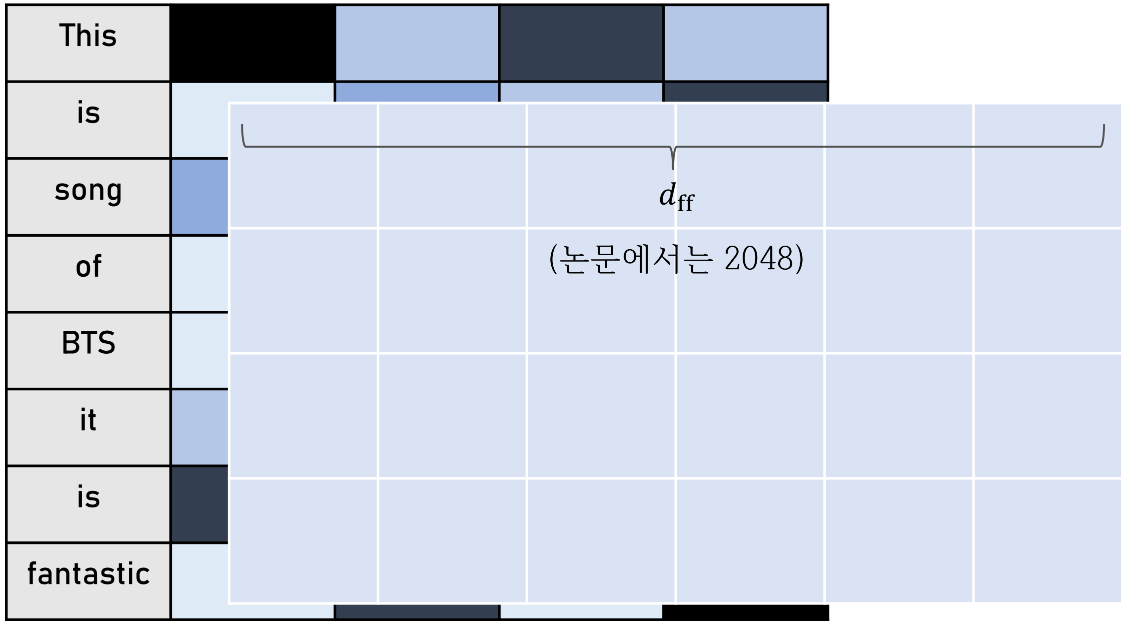


Transformer Feed Forward



② ReLU ① Linear Formation ③ Linear Formation

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



Code

https://colab.research.google.com/drive/19sECAn3d0993MxckIVmngCXwE0AjbmRk?usp=share_link