Attention is all you need

NIPS' 17

Jung Soo Lee
19.11.26

Attention is all you need NIPS’ 17

- Novelty : first transduction model entirely based on attention
- 2 main contributions:
  - 1) parallelization
    - Attention on each position (X need to wait previous hidden state like RNN)
  - 2) reduction of computation

Attention is all you need NIPS' 17

Model Architecture

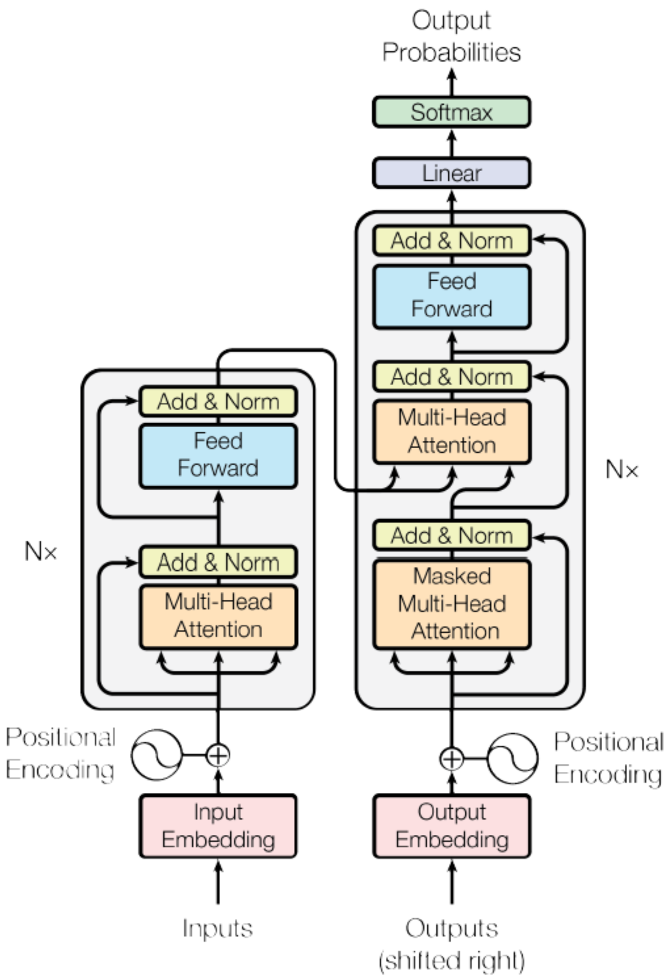


Figure 1: The Transformer - model architecture.

- Encoder-decoder structure
  - Autoregressive (use all previous info)
  - x1,x2,…,xn -> z1,z2,…zn -> y1,y2…,ym
- Encoder
  - Multi-Head Attention + feed forward
  - N layers
- Decoder
  - Multi-Head Attention + feed forward + linear & softmax
  - N layers
- Attention -> Key / Value / Query

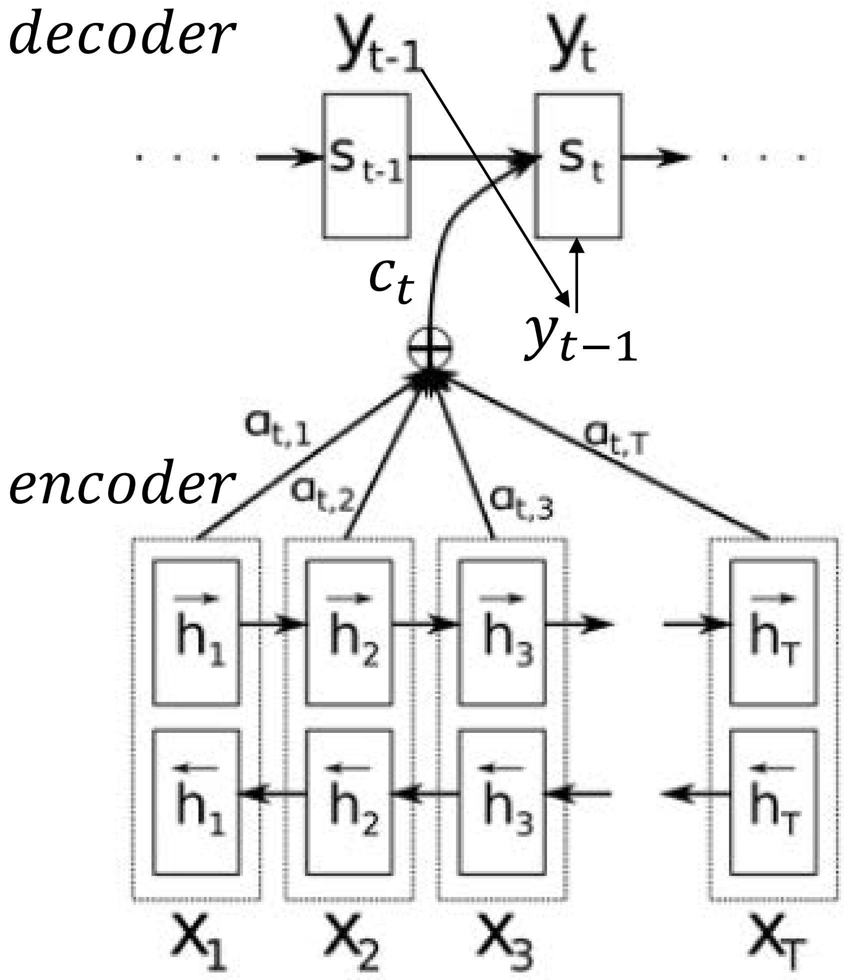
Attention is all you need NIPS' 17

Attention

- Seq2seq model -> long term dependency problem
  - Solve: context vector can't deliver all information
- Assumption
  - Encoder time t vector similar with Decoder time $t-1$ vector
  - Similar -> increased value of each position in a vector

Attention is all you need NIPS' 17

Attention

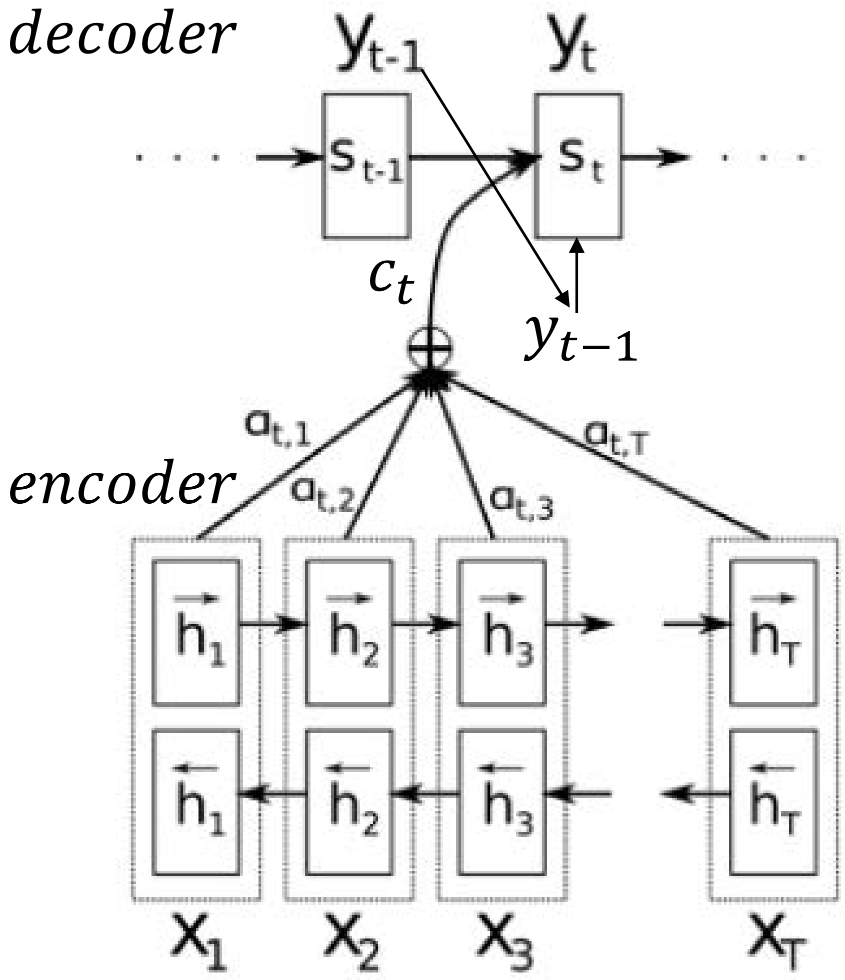


- x_t: input from encoder in time t
- y_t: output from decoder in time t
- h_n : encoder nth hidden state vector
- s_t: decoder hidden state vector in time t
- $a_{t,n}$: n^{th} weight in time t (=attention vector)
  - Which vector to look into
- c_t: context vector in time t
- T = length of the sequence
- $s_t = f(s_{t-1}, y_{t-1}, c_t)$
- Assumption -> training is finished

https://nlpstudynote.tistory.com/18

Attention is all you need NIPS' 17

Attention

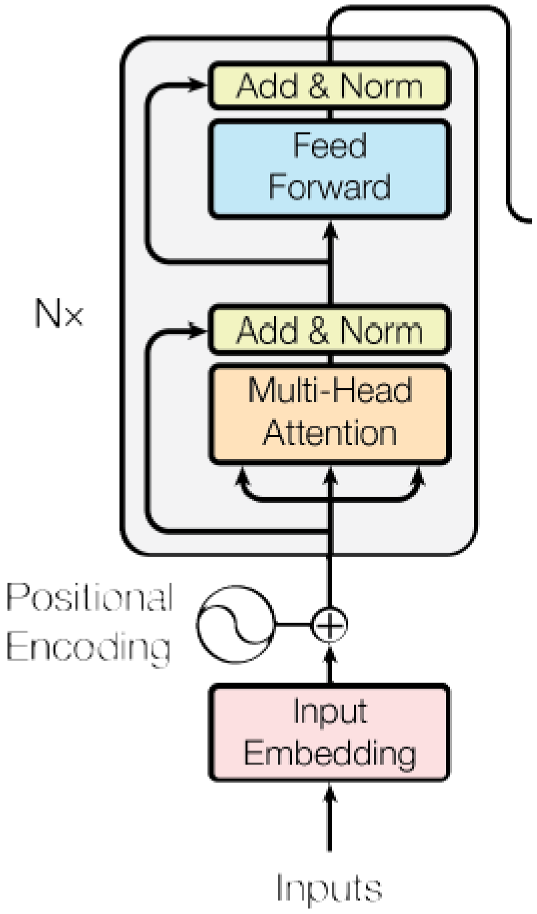


- $s_t = f(s_{t-1}, y_{t-1}, c_t)$
- $c_t = \sum_{i=1}^{T} a_{t,i}\, h_i$
  - weighted sum of attention vector
- $a_{t,i}$ = softmax(e_{ti})
- e_{ti} = score value
  - a(s_{t-1}, h_i)
  - a : alignment model (extracts similarity)
  - Various methods for alignment model
  - a : $v_a^T \tanh(W_a[s_{t-1}, h_i])$
    - V, W -> parameters
    - Earned during training
  - a : $[s_{t-1}^T h_i] \rightarrow e_{ti} = [s_{t-1}^T h_1, s_{t-1}^T h_2, \ldots, s_{t-1}^T h_N]$

https://nlpstudynote.tistory.com/18

Attention is all you need NIPS' 17

Transformer – Encoder



- Positional Encoding
- Multi-head self attention & position-wise fully connected feed-forward network
- Layer normalization
- Residual connection for two sublayers
  - x+Sublayer(x)
- Stacked N layers

Attention is all you need NIPS' 17

Positional Encoding

$$PE_{(pos,2i)} = \sin(pos/1000^{2i/d_{model}})$$
$$PE_{(pos,2i+1)} = \cos(pos/1000^{2i/d_{model}})$$

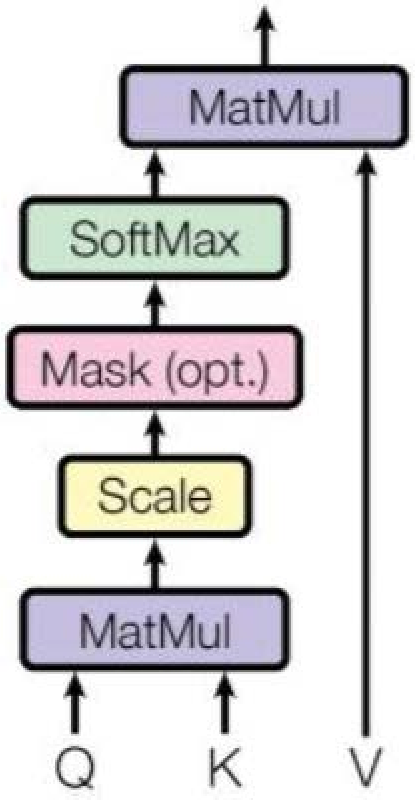
- Word Embedding -> positional information
- Positional encoding vector + embedding vector (same dimension)
- Pos: position / i : dimension
- Function: sinusoidal function with cycle : $1000^{2i/d_{model}} \cdot 2\pi$

- $PE_{pos} = [\cos(pos/1), \sin(pos/1000^{2/d_{model}}), \ldots, \sin(pos/1000^{2i/d_{model}})]$
- Same word have different value in different position
- Linear Function

https://pozalabs.github.io/transformer/

Attention is all you need NIPS' 17

Scaled dot product attention



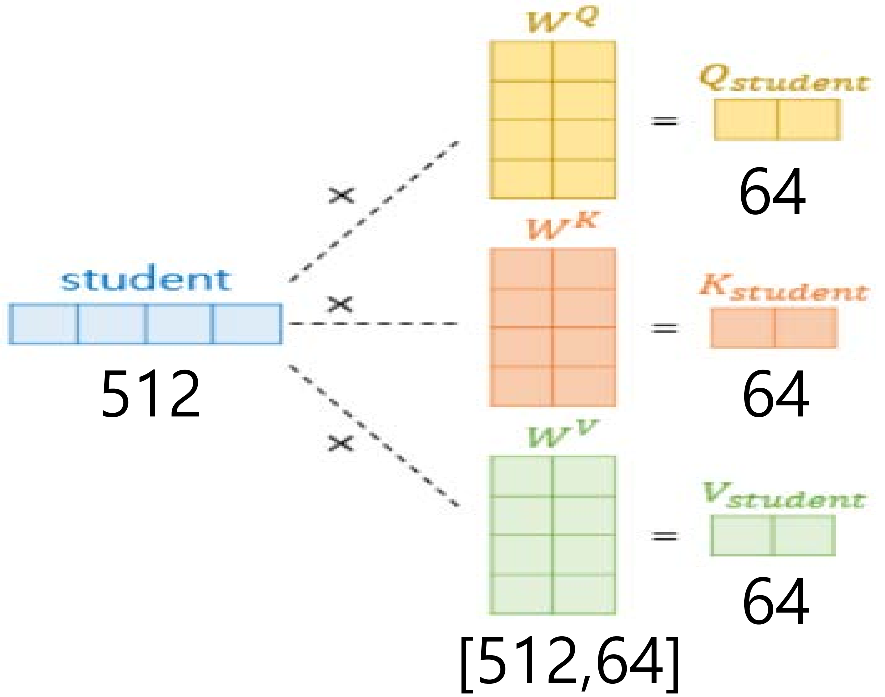
- $Attention(Q,K,V) = softmax(\mathrm{Q}K^T/\sqrt{d_k})\mathrm{V}$
  - d_k = dimension of K
- Key: hidden state vector of a word
- Value: hidden state vector of a word
- Query: hidden state vector of a word (current-word)
- dot product -> similarity between K & Q
- Scaling -> Softmax: big value have minimal gradient
- Softmax * V -> Attention(Q,K,V) increases as value and query are similar
- Faster than additive attention

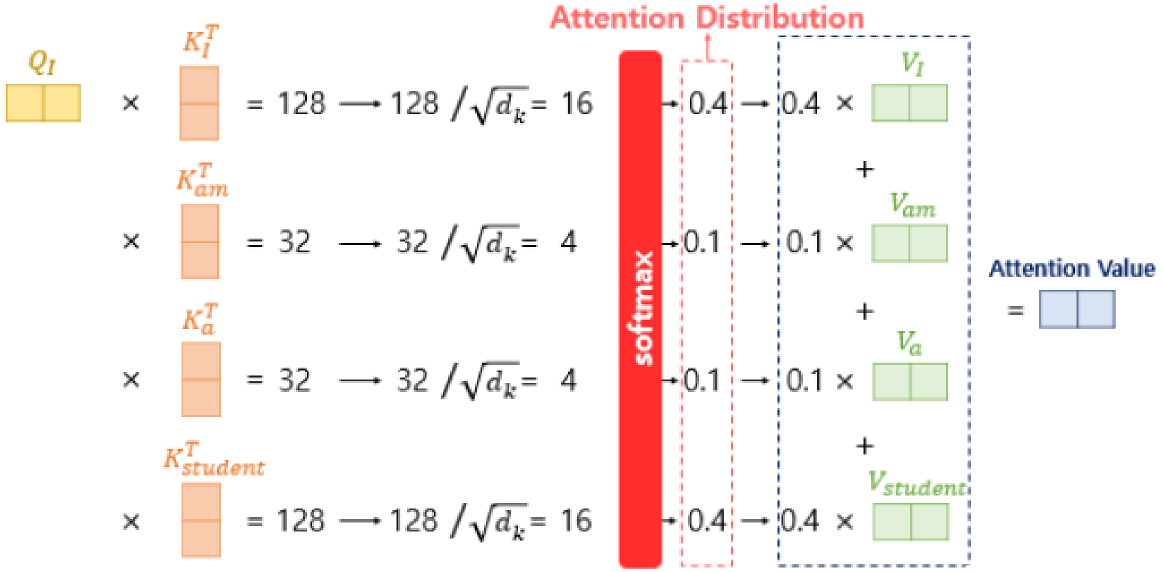
https://pozalabs.github.io/transformer/

Attention is all you need NIPS' 17

Scaled dot product attention – K, V, Q

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





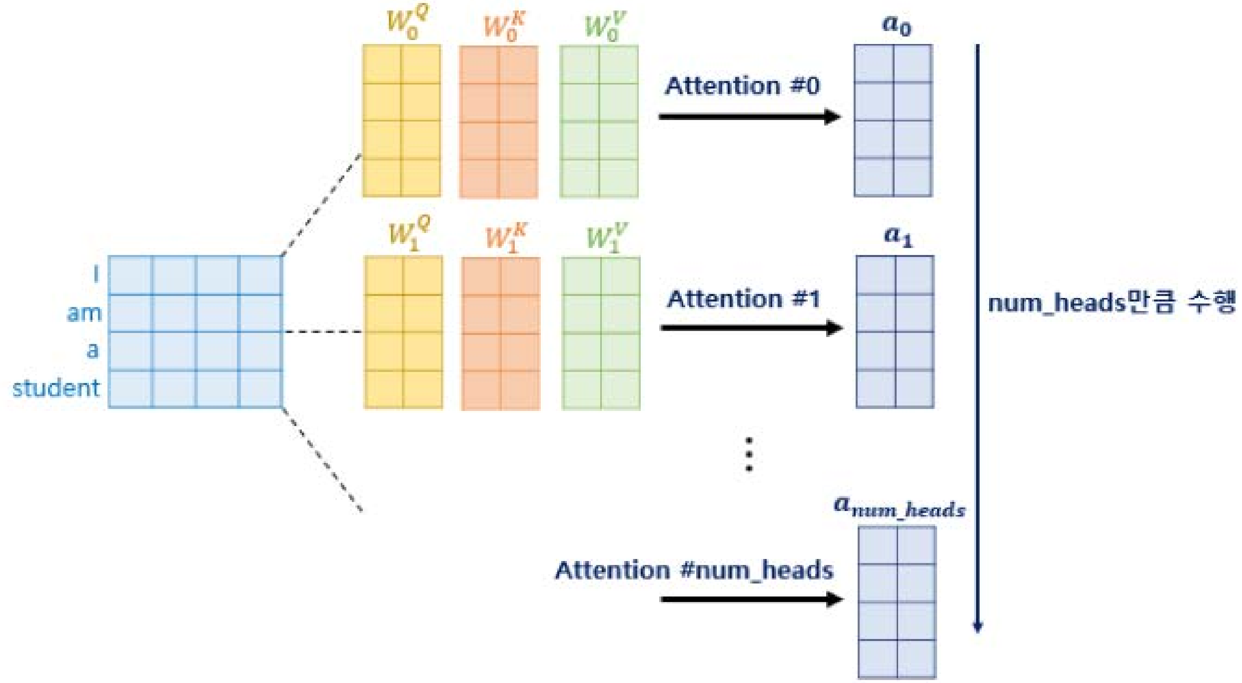
- d_{model} = dimension of word embedding = 512 (normally length of longest sentence)
- $d_k = d_V = d_Q = d_{model}$ / num_heads
  - Num_heads = 8

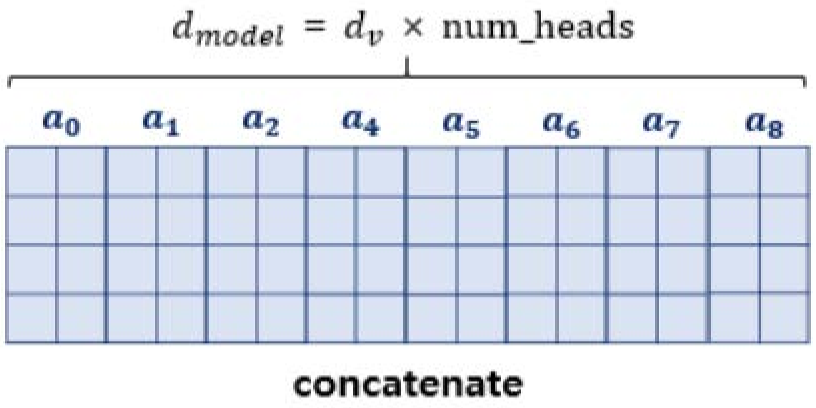
https://wikidocs.net/31379

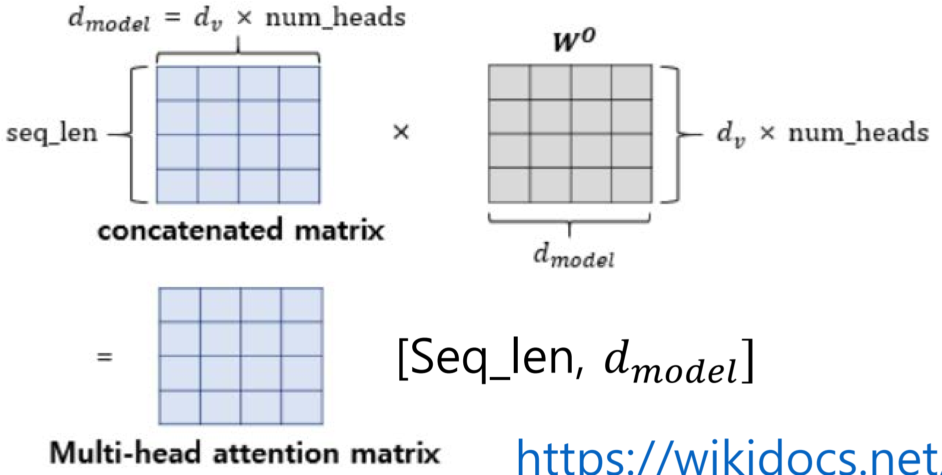
Attention is all you need NIPS' 17

Multi-head Attention –K,V,Q

$$Multi - head\ Attention = Concat(head1, head2, \ldots, headh)\ W^O$$
$$headi\ = \text{Attention}(QWi^Q, KWi^K, VWi^V)$$





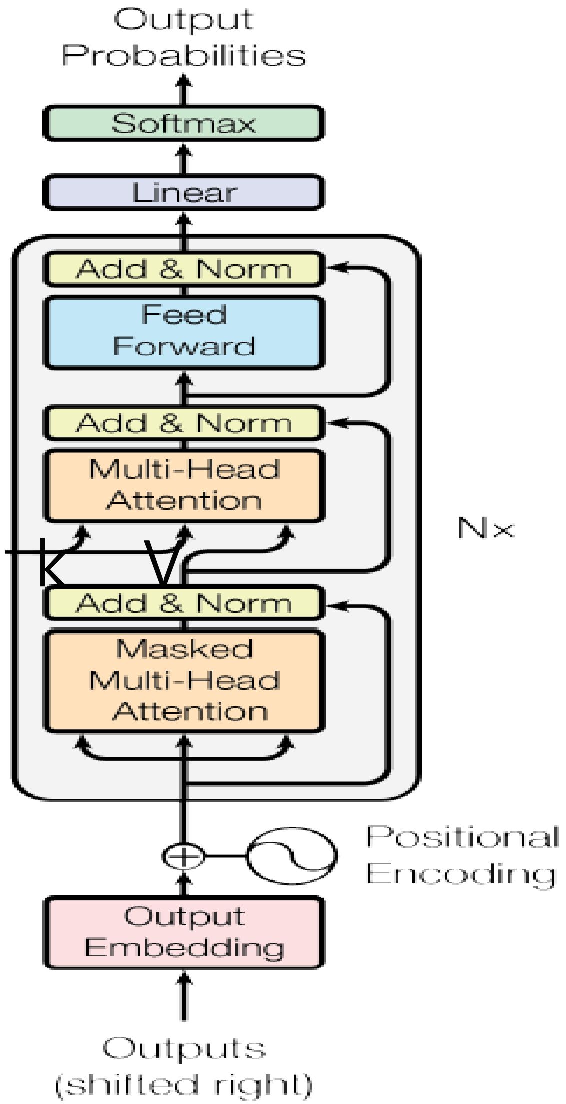


[Seq_len, d_{model}]

https://wikidocs.net/31379

Attention is all you need NIPS' 17

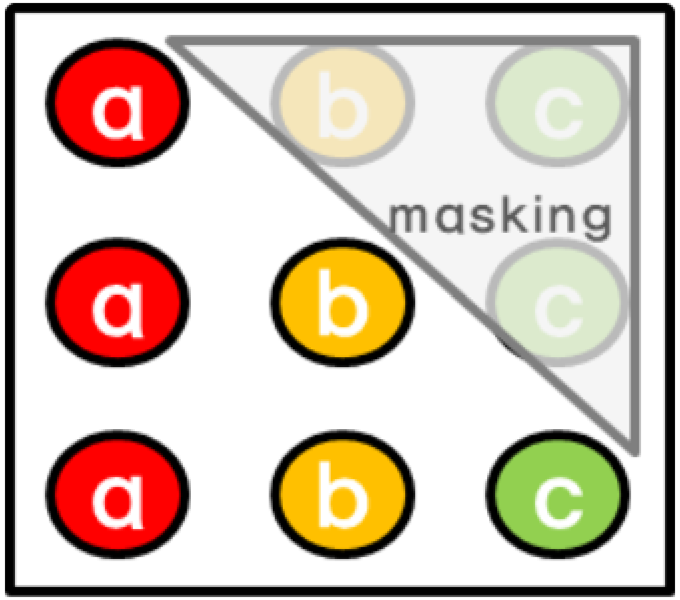
Transformer – Decoder



- Positional Encoding
- Masking
- Multi-head self attention & position-wise fully connected feed-forward network
  - 2^{nd} : from encoder output (key, value)
- Residual connection for three sublayers
  - x+Sublayer(x)
- Layer normalization
- Linear / Softmax

Attention is all you need NIPS' 17

Masking



1 Attention until a -> don't look into b,c

2 Attention until a,b -> don't look into c

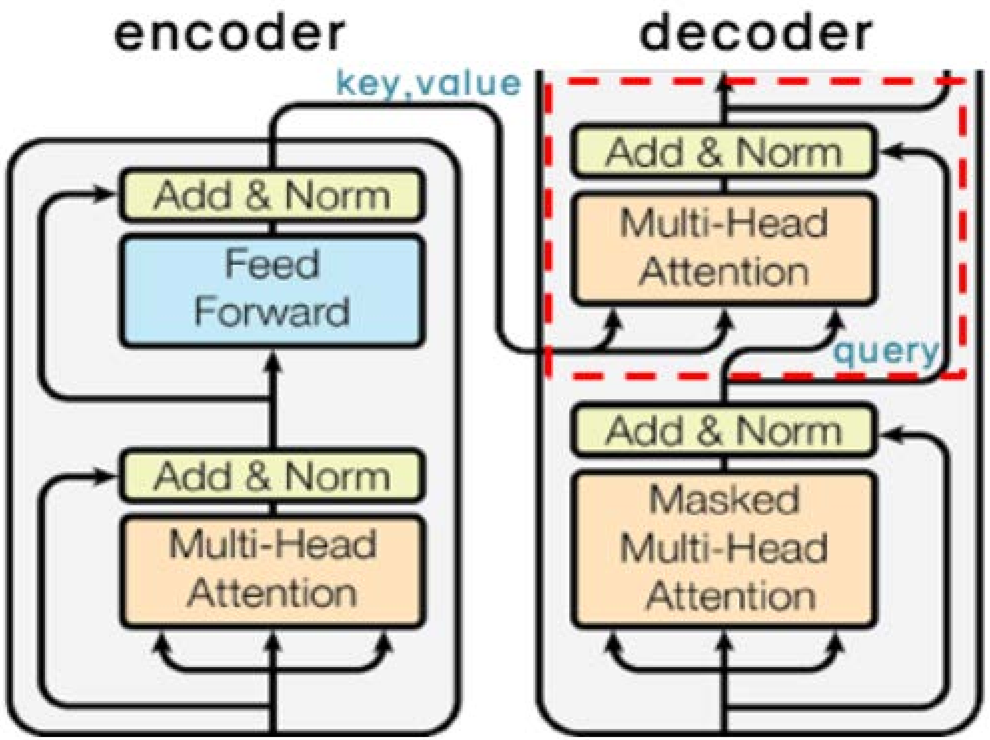
3 Attention until a,b,c

Masked input value = $-\infty$

https://pozalabs.github.io/transformer/

Attention is all you need NIPS' 17

Encoder – Decoder



- Query from decoder attention layer -> attention until i^{th} position due to masking out
- Can give attention to all positions

https://pozalabs.github.io/transformer/

Attention is all you need NIPS’ 17

Regularization

1. Dropout to output of each sub-layer
2. Dropout to output to positional encoding
3. Label smoothing

Strength of Self Attention

1. Computational complexity
2. Parallelization
3. Shorten path length

Attention is all you need NIPS' 17

Result

- Task: Machine Translation (WMT 2014 English-to-German translation & WMT 2014 English-to-French translation task)

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		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [15]	23.75			
Deep-Att + PosUnk [32]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [31]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [8]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [26]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [32]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [31]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [8]	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**	**41.0**	$2.3 \cdot 10^{19}$	