NATURAL LANGUAGE PROCESSING

LECTURE 9: Transformer

Contents

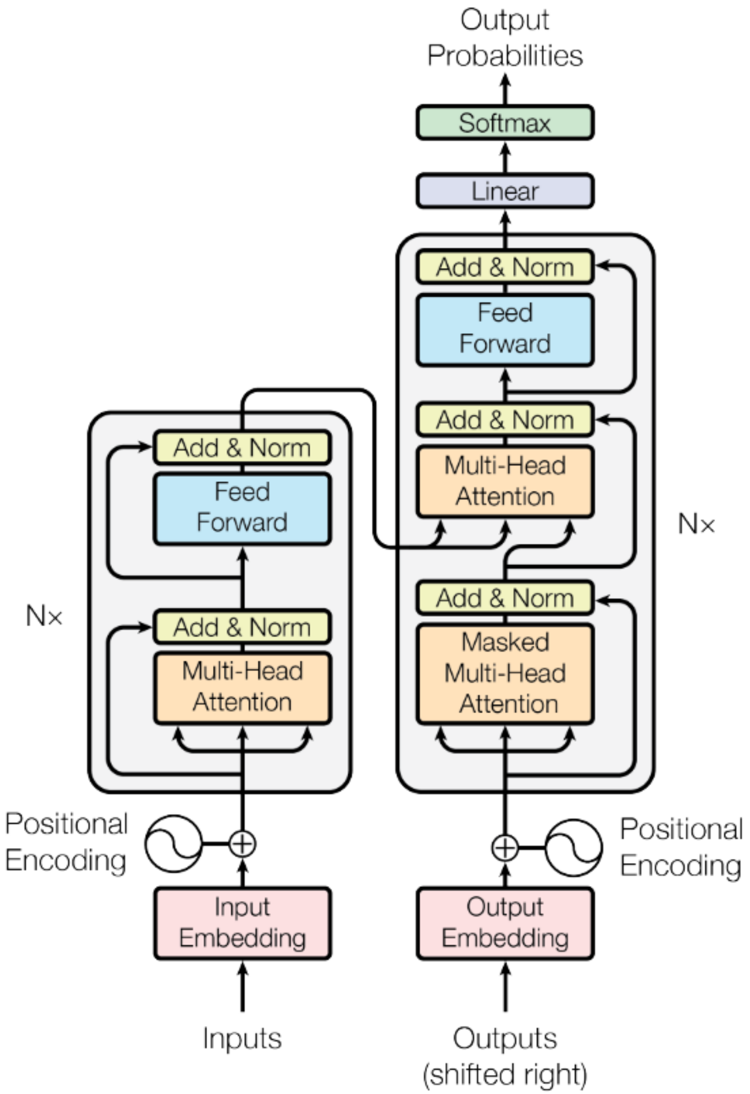
Transformer (self-attention model)

- Scaled dot-product attention, multi-head attention, …

Contextual representations

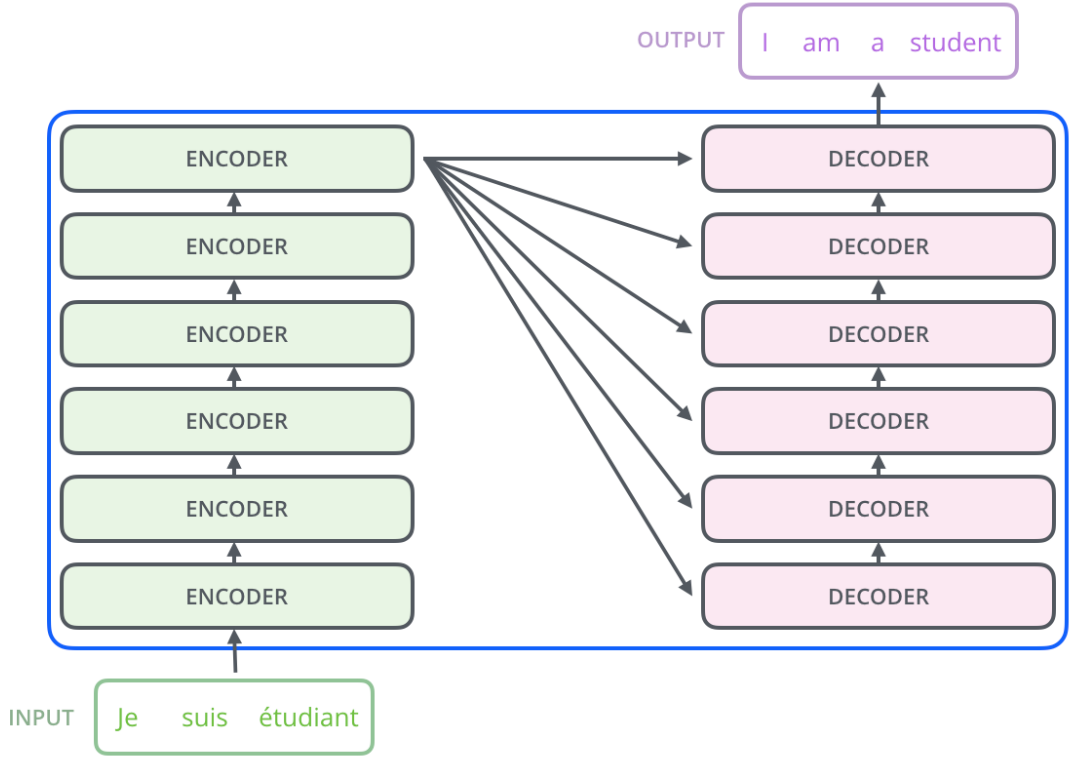
- ELMo, CoVe, ULMFiT, …

Transformer: High-level view

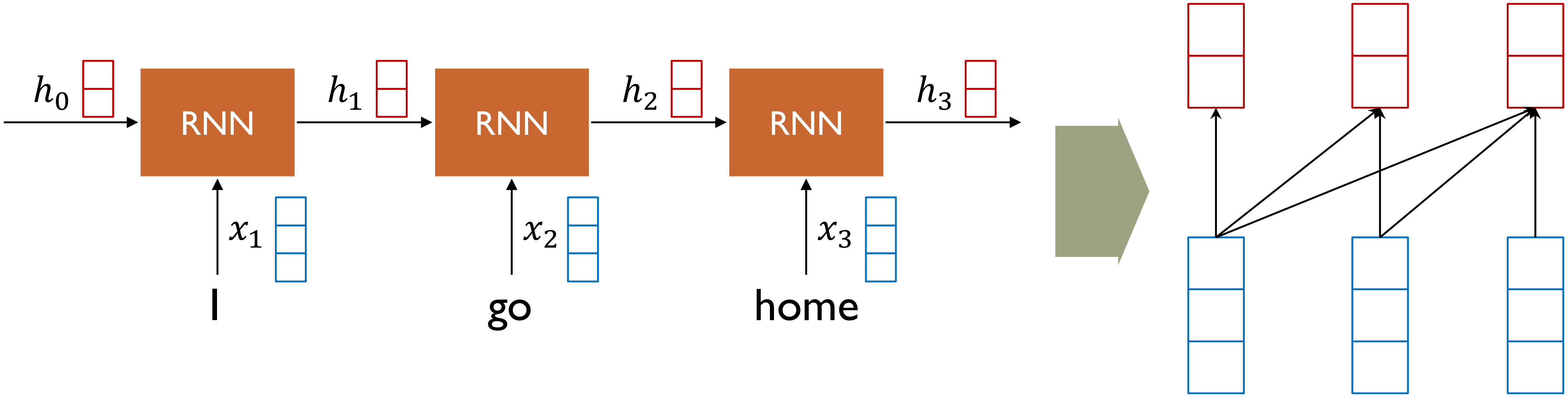


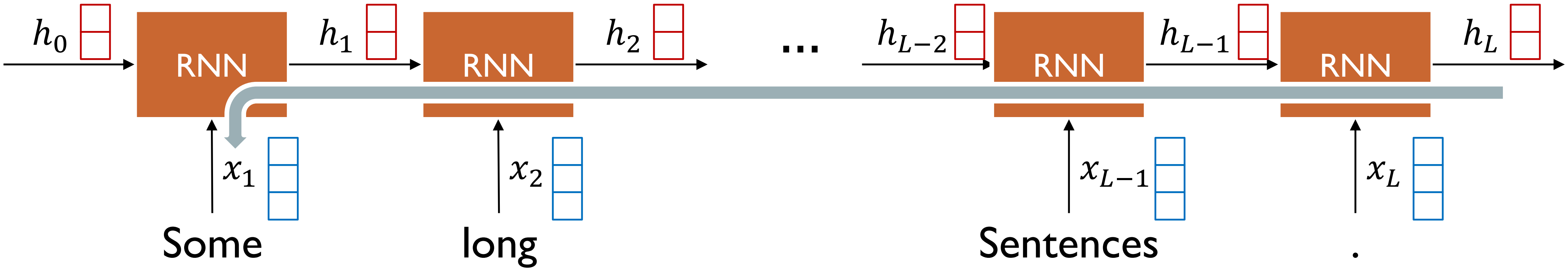
Attention is all you need, NeurIPS'17

- No more RNN or CNN modules

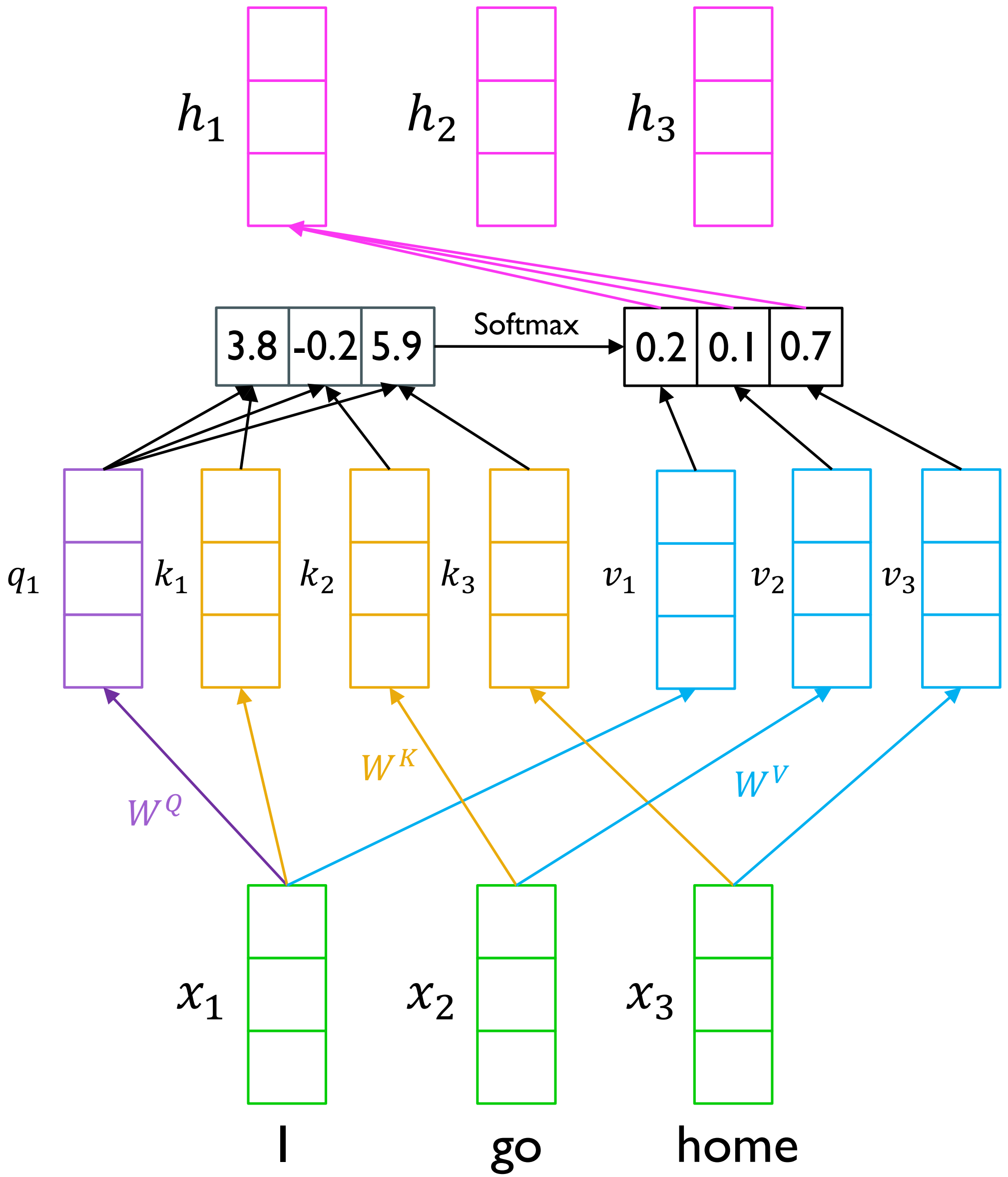


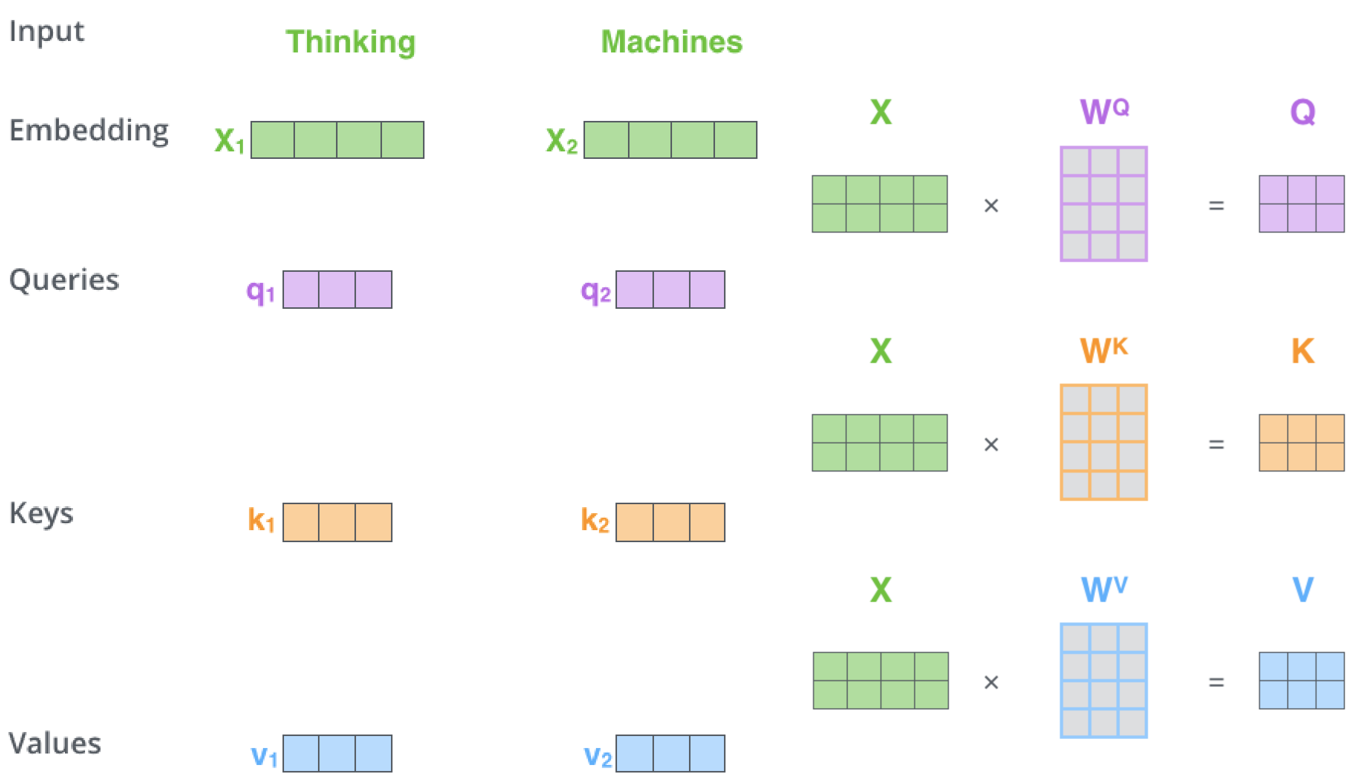
RNN: Long-term dependency





Transformer: Long-term dependency





Attention Is All You Need, NeurIPS'17

Transformer: Scaled dot-product attention

- Inputs: a query q and a set of key-value (k, v) pairs to an output
- Query, keys, values, and output are all vectors

- Output is weighted sum of values
- Weight of each value is computed by an inner product of query and corresponding key
- Queries and keys have same dimensionality d_k, and dimensionality of value is d_v

$$A(q, K, V) = \sum_i \frac{\exp(q \cdot k_i)}{\sum_j \exp(q \cdot k_j)} v_i$$

Transformer: Scaled dot-product attention

- When we have multiple queries q, we can stack them in a matrix Q:

$$A(q, K, V) = \sum_i \frac{\exp(q \cdot k_i)}{\sum_j \exp(q \cdot k_i)} v_i$$

- Becomes:

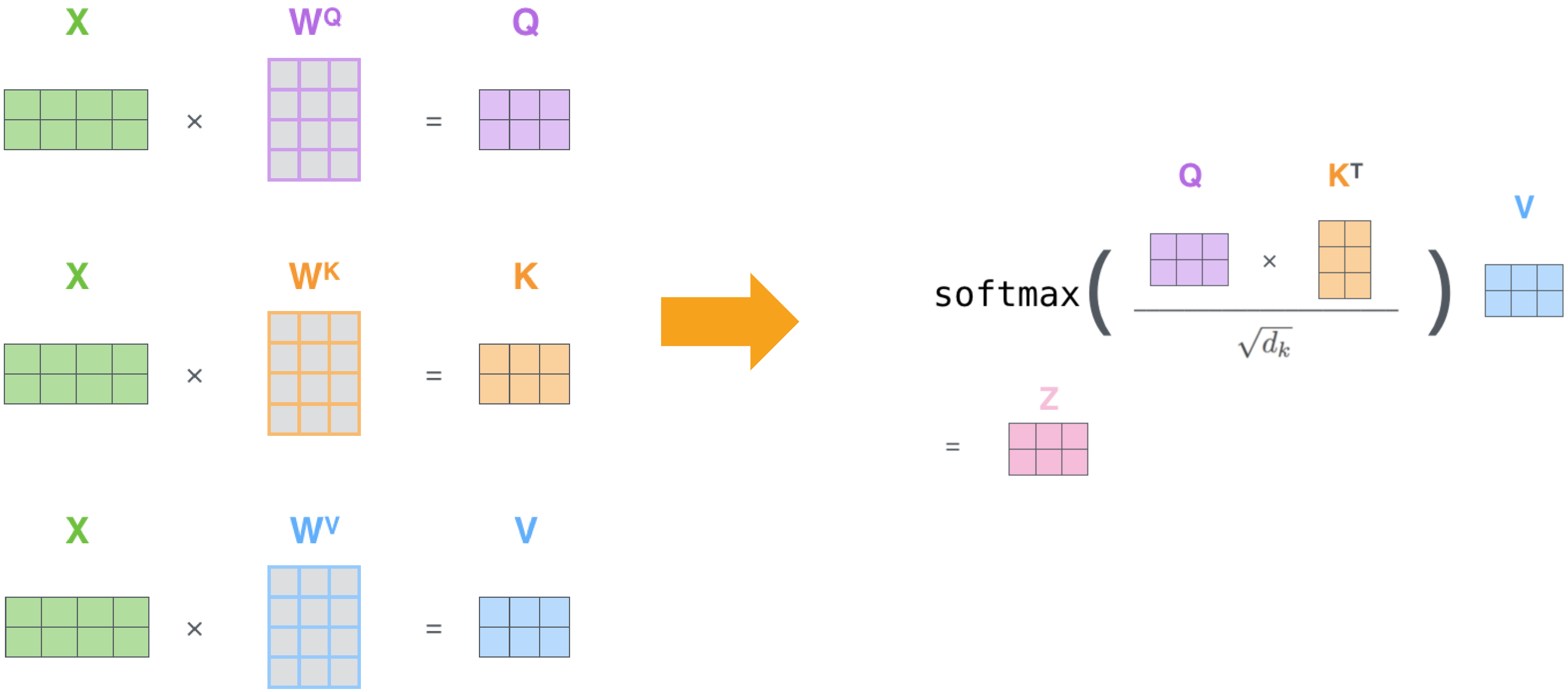
$$A(Q, K, V) = \mathrm{softmax}(QK^T)V$$

$$(|Q| \times d_k) \times (d_k \times |K|) \times (|K| \times d_v) = (|Q| \times d_v)$$

Row-wise softmax

Transformer: Scaled dot-product attention

- Example (from **illustrated transformer**)



http://jalammar.github.io/illustrated-transformer/

Transformer: Scaled dot-product attention

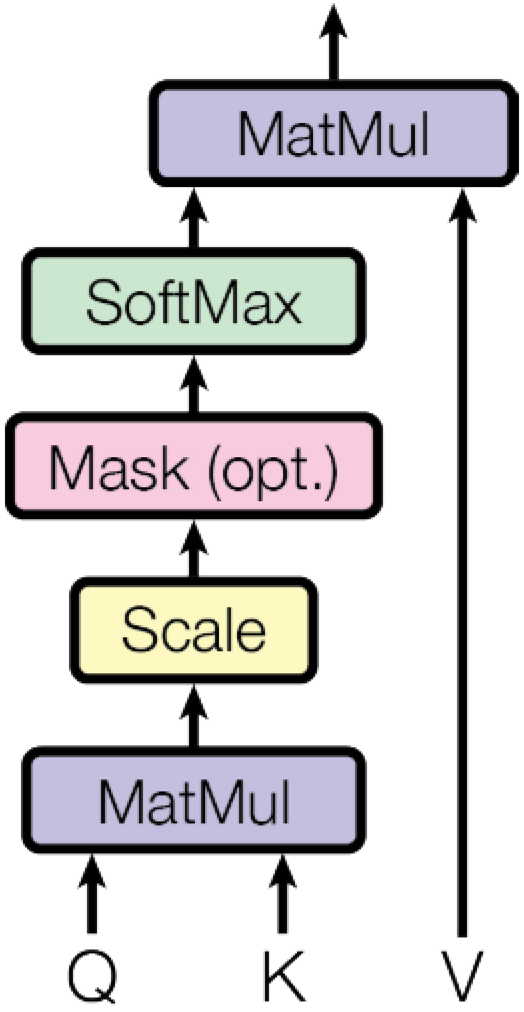
Problem

- As d_k gets large, the variance of $q^T k$ increases
- Some values inside the softmax get large
- The softmax gets very peaked
- Hence its gradient gets smaller

Solution

- Scaled by the length of query / key vectors:

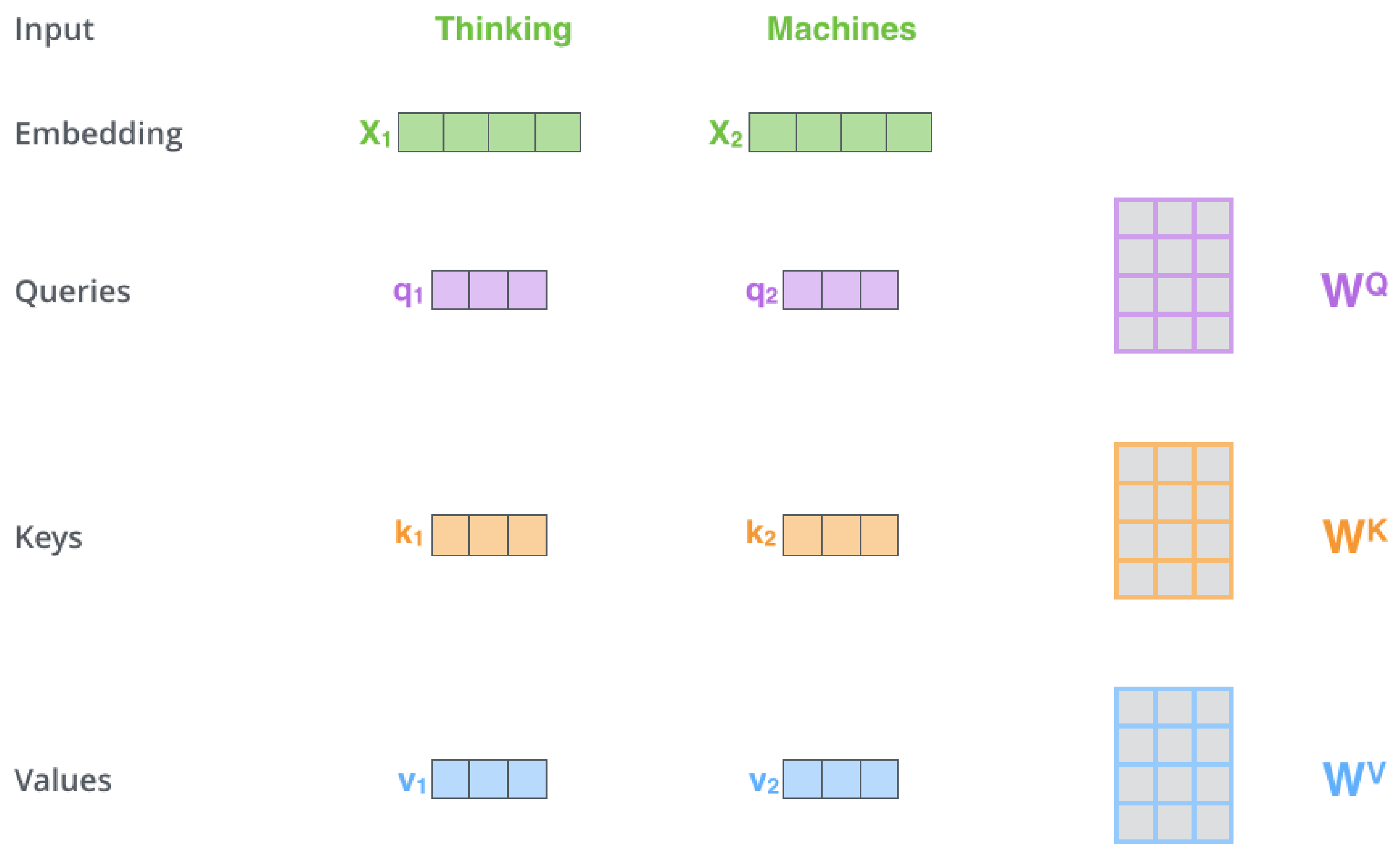
$$A(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



Attention Is All You Need, NeurIPS'17

Transformer: Scaled dot-product attention

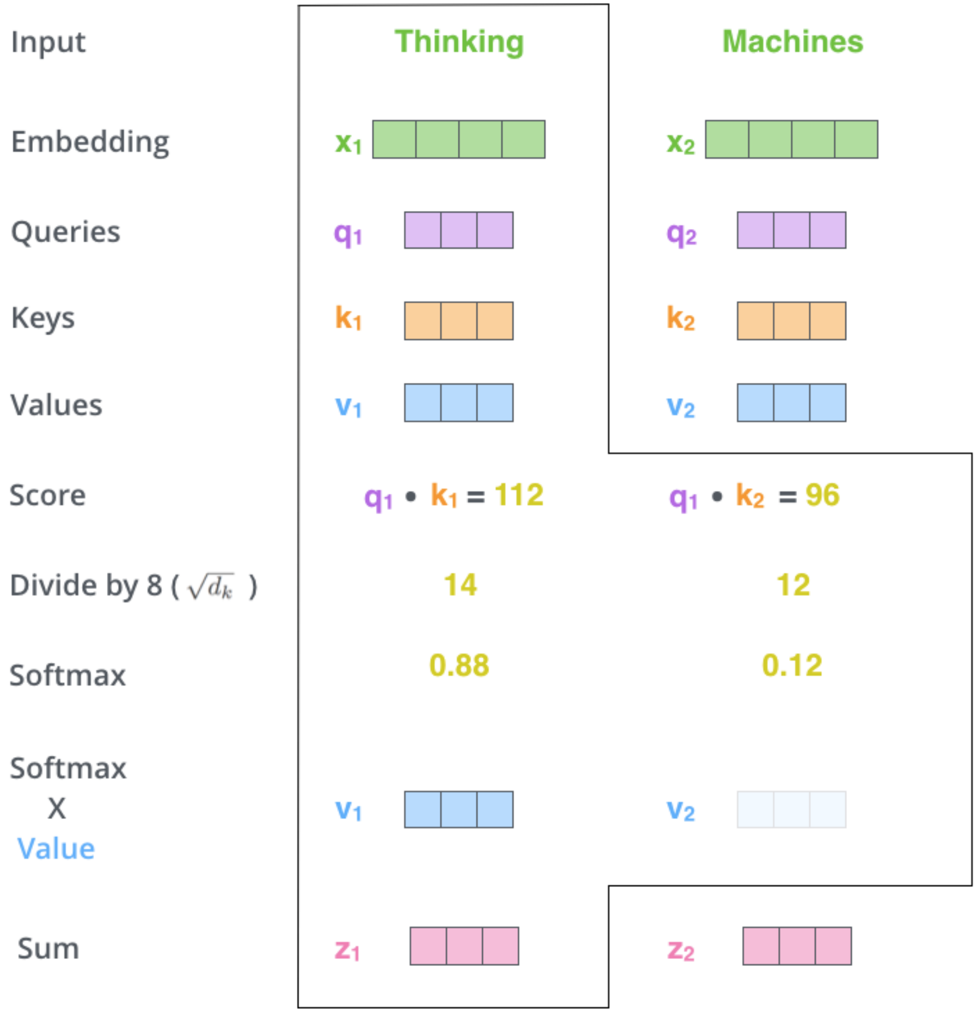
- Example (from **illustrated transformer**)



http://jalammar.github.io/illustrated-transformer/

Transformer: Scaled dot-product attention

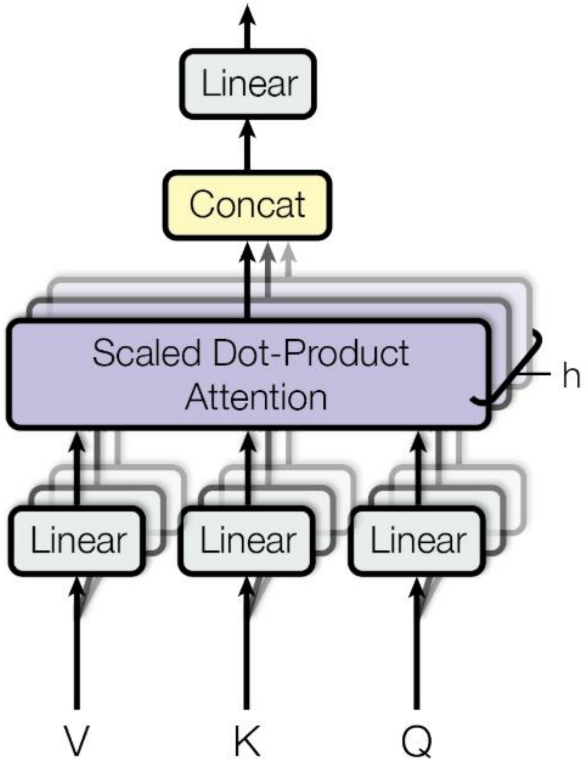
- Example (from **illustrated transformer**)



http://jalammar.github.io/illustrated-transformer/

Transformer: Multi-head attention

- The input word vectors could be the queries, keys and values
- In other words, the word vectors themselves select each other
- Problem: Only one way for words to interact with one another
- Solution: Multi-head attention maps Q, K, V into the h number of lower-dimensional spaces via W matrices
- Then apply attention, then concatenate outputs and pipe through linear layer

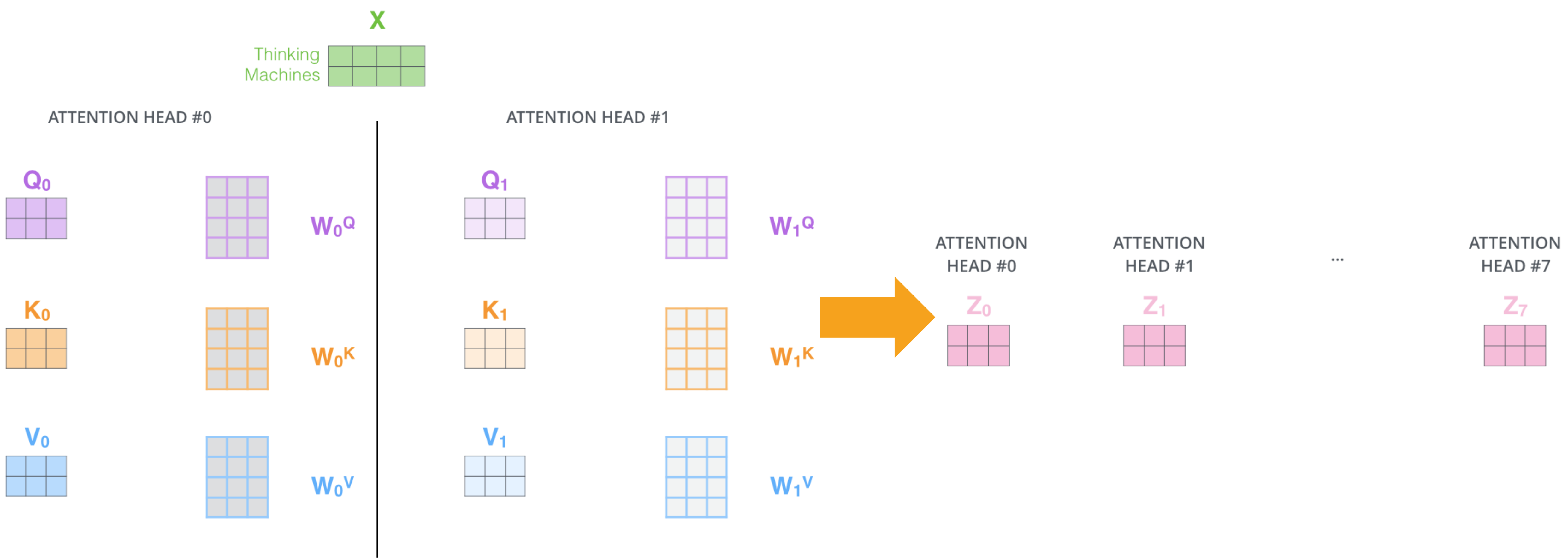


$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head}_1, \dots, \mathrm{head}_h) W^O$$

$$\text{where } \mathrm{head}_i = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Transformer: Multi-head attention

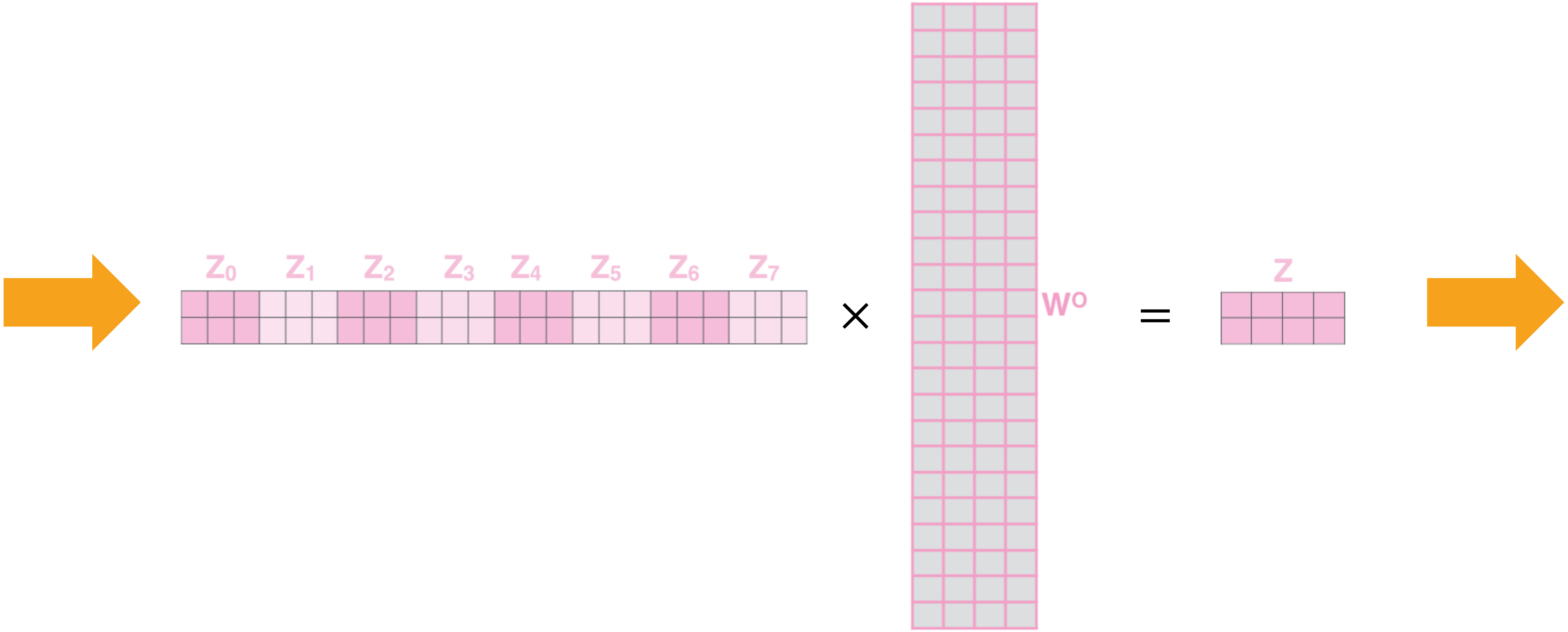
- Example (from **illustrated transformer**)



http://jalammar.github.io/illustrated-transformer/

Transformer: Multi-head attention

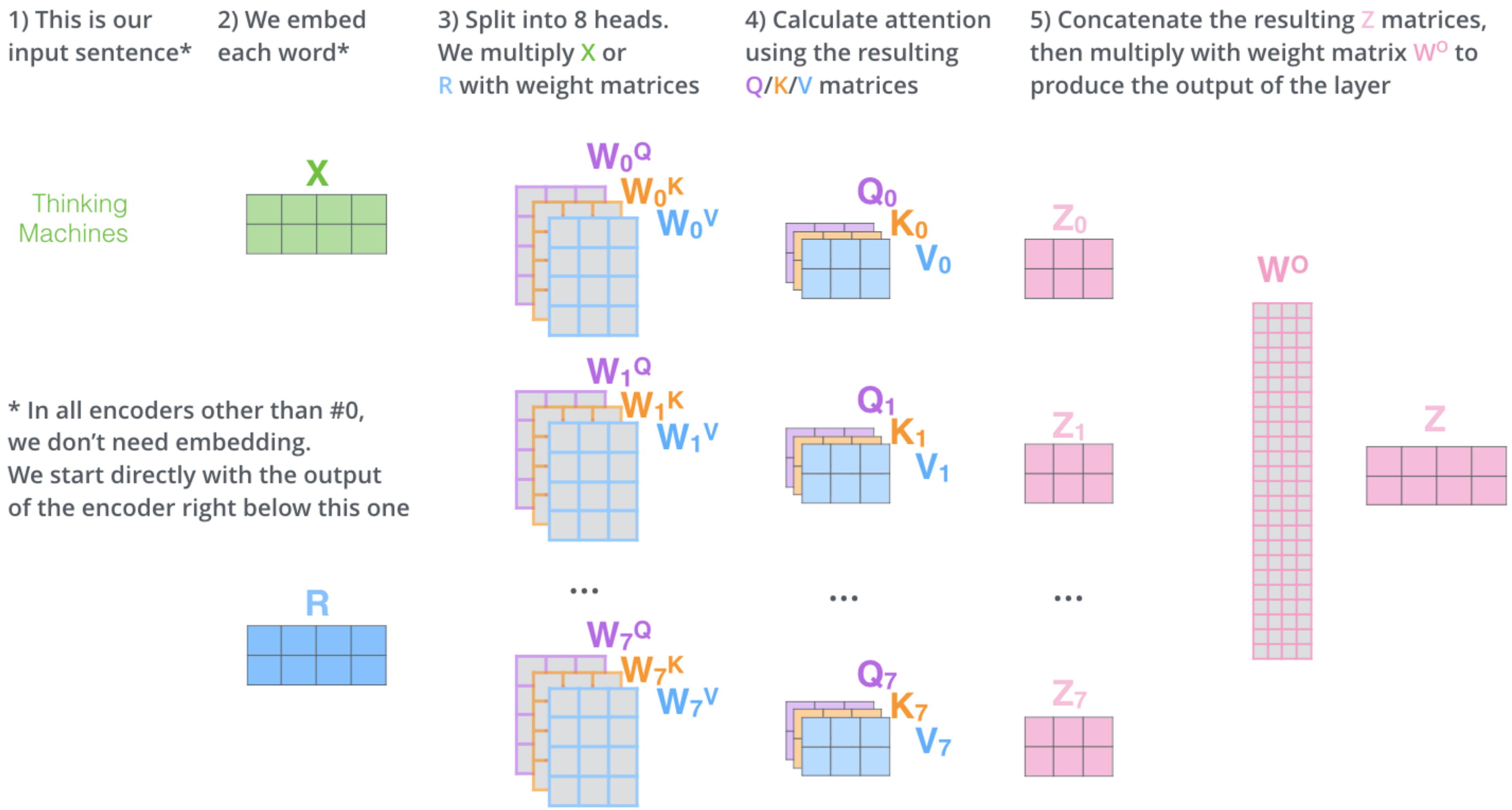
- Example (from **illustrated transformer**)



http://jalammar.github.io/illustrated-transformer/

Transformer: Multi-head attention

- Example (from **illustrated transformer**)



http://jalammar.github.io/illustrated-transformer/

Transformer: Multi-head attention

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

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, NeurIPS'17

Transformer: Block based model

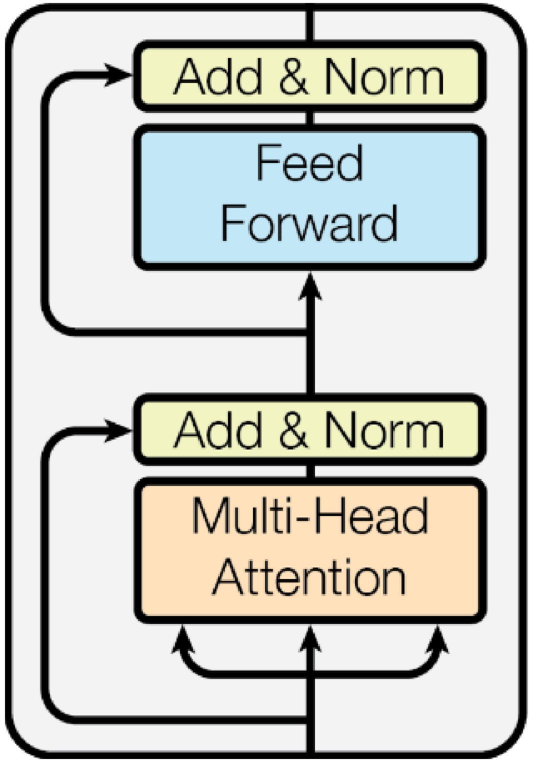
Each block has two **sub-layers**

- Multi-head attention
- Two-layer feed-forward NN (with ReLU)

Each of these two steps also has

- Residual connection and layer normalization:

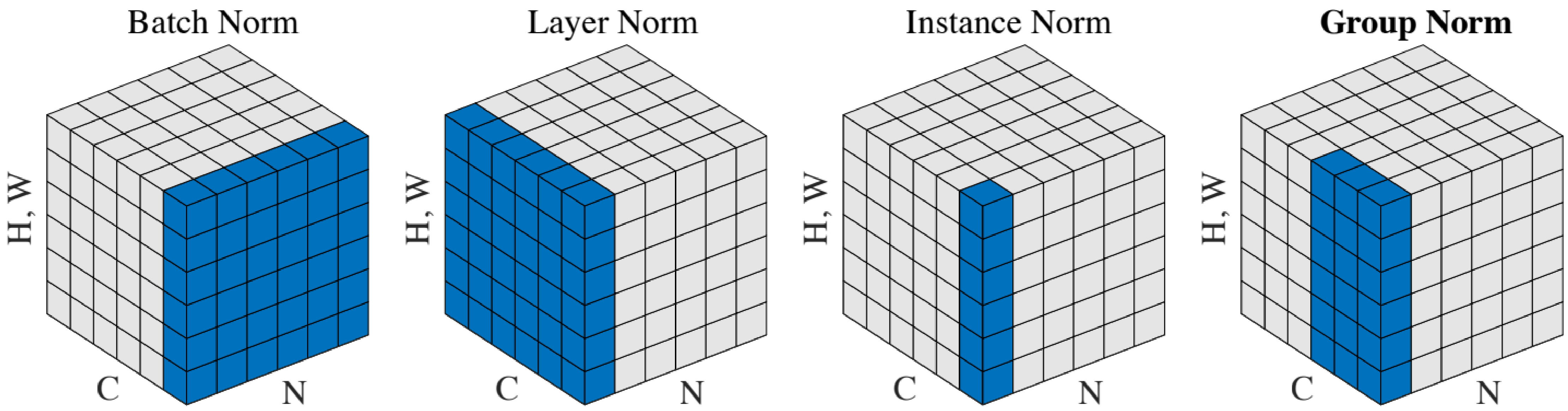
$$LayerNorm(x + sublayer(x))$$



Layer Normalization

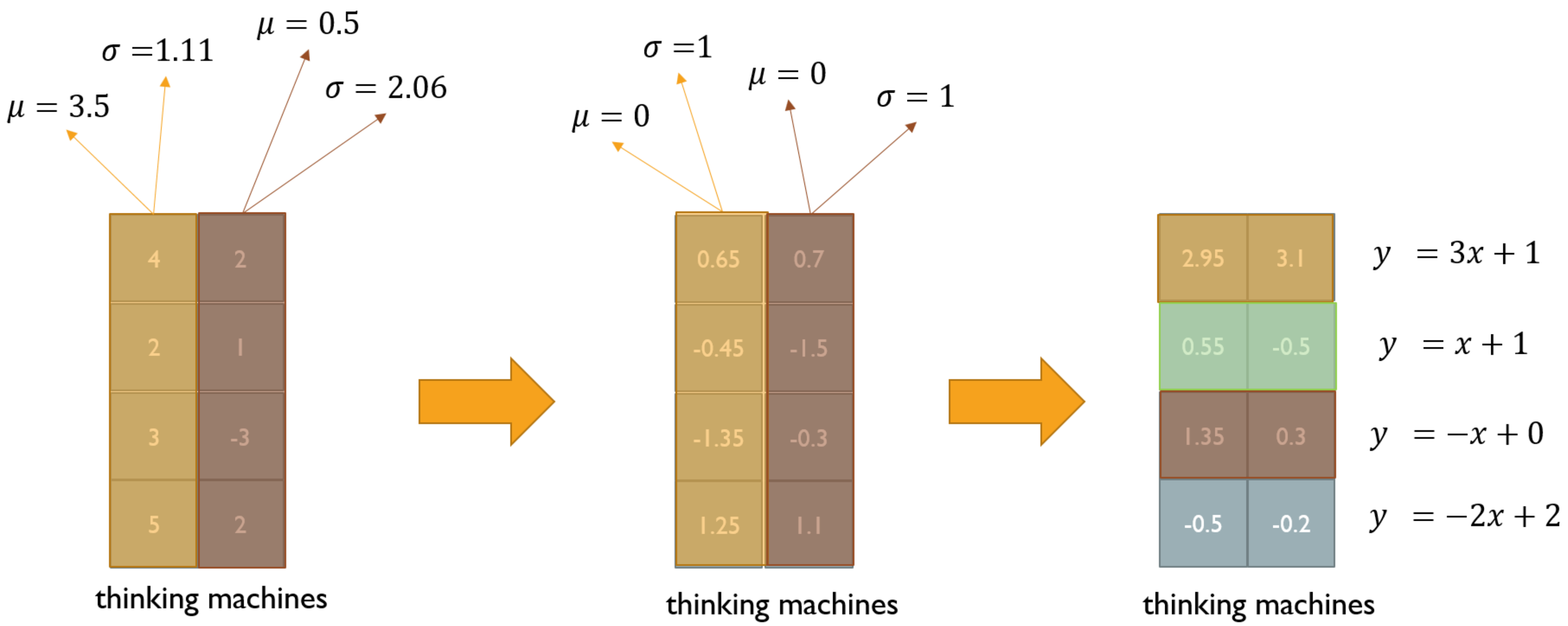
- Layer normalization changes input to have zero mean and unit variance, per layer and per training point (and adds two more parameters)

$$\mu^l = \frac{1}{H}\sum_{i=1}^{H} a_i^l, \qquad \sigma^l = \sqrt{\frac{1}{H}\sum_{i=1}^{H}(a_i^l - \mu^l)^2}, \qquad h_i = f\left(\frac{g_i}{\sigma_i}(a_i - \mu_i) + b_i\right)$$



Group Normalization, ECCV'18

Layer Normalization



Layer normalization consists of two steps:

- Normalization of each word vectors to have mean of zero and variance of one.
- Affine transformation of each sequence vector with learnable parameters.

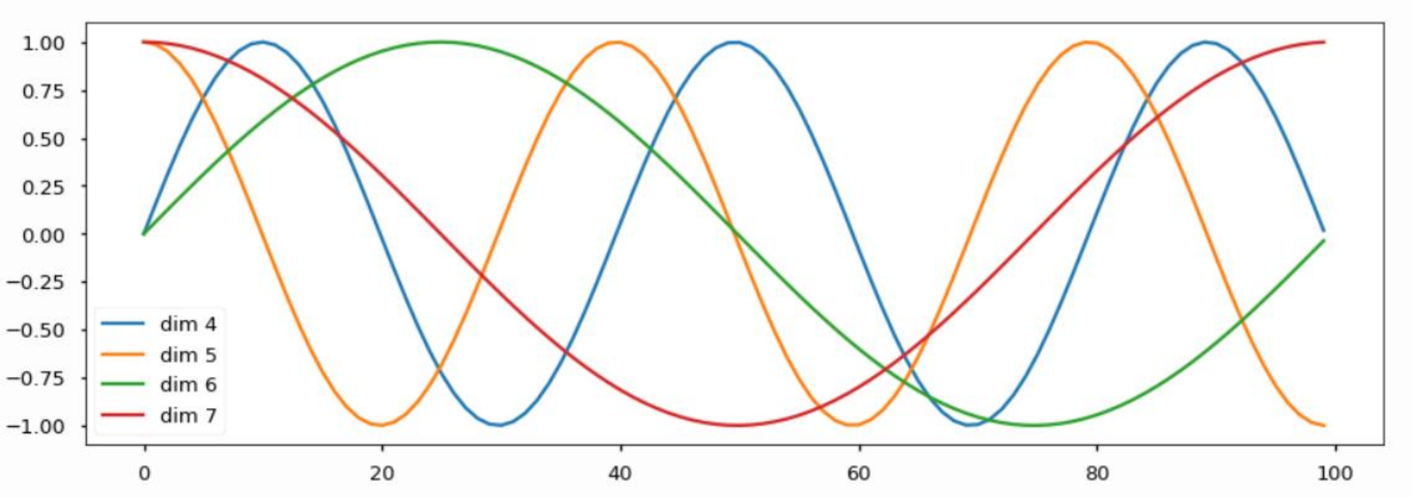
Transformer: Positional Encoding

- Use sinusoidal functions of different frequencies

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

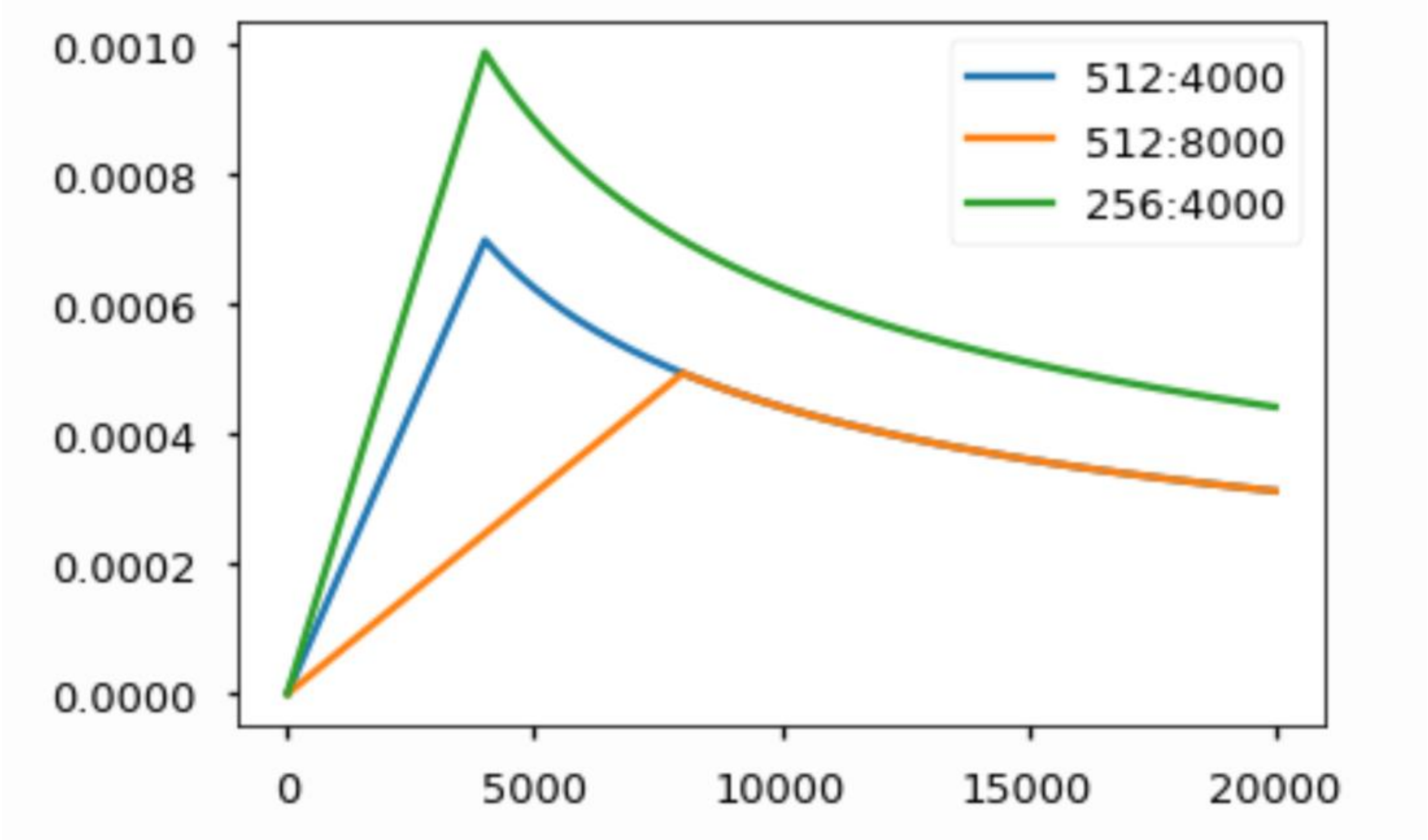
- Easily learn to attend by relative position, since for any fixed offset k, $PE_{(pos+k)}$ can be represented as linear function of $PE_{(pos)}$

- Another positional encoding is okay to use (e.g., positional encoding in ConvS2S)



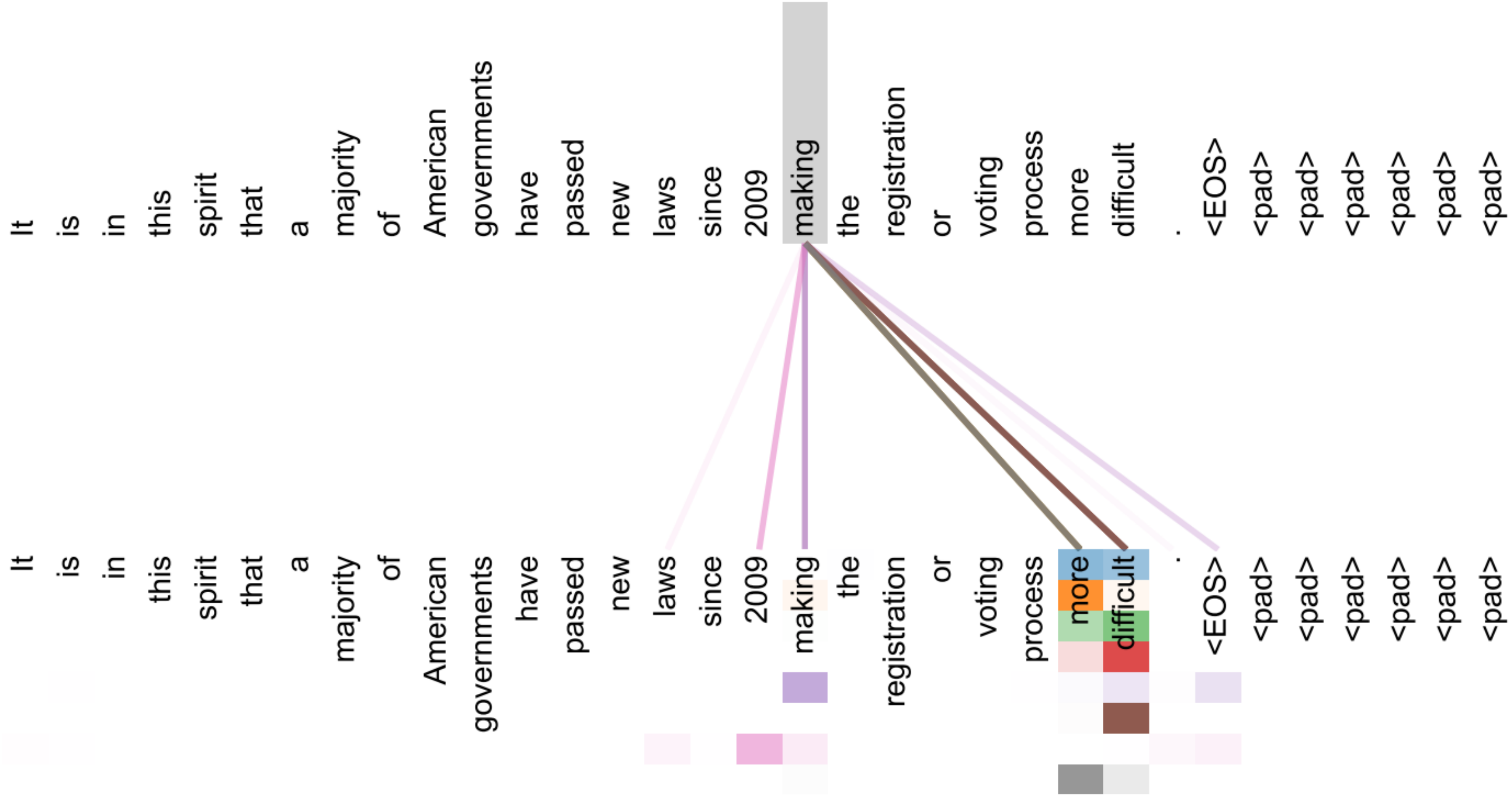
Transformer: Warm-up Learning Rate Scheduler

$$lrate = d_{model}^{-0.5} \cdot \min(\#step^{-0.5}, \#step \cdot warmup\_steps^{-1.5})$$



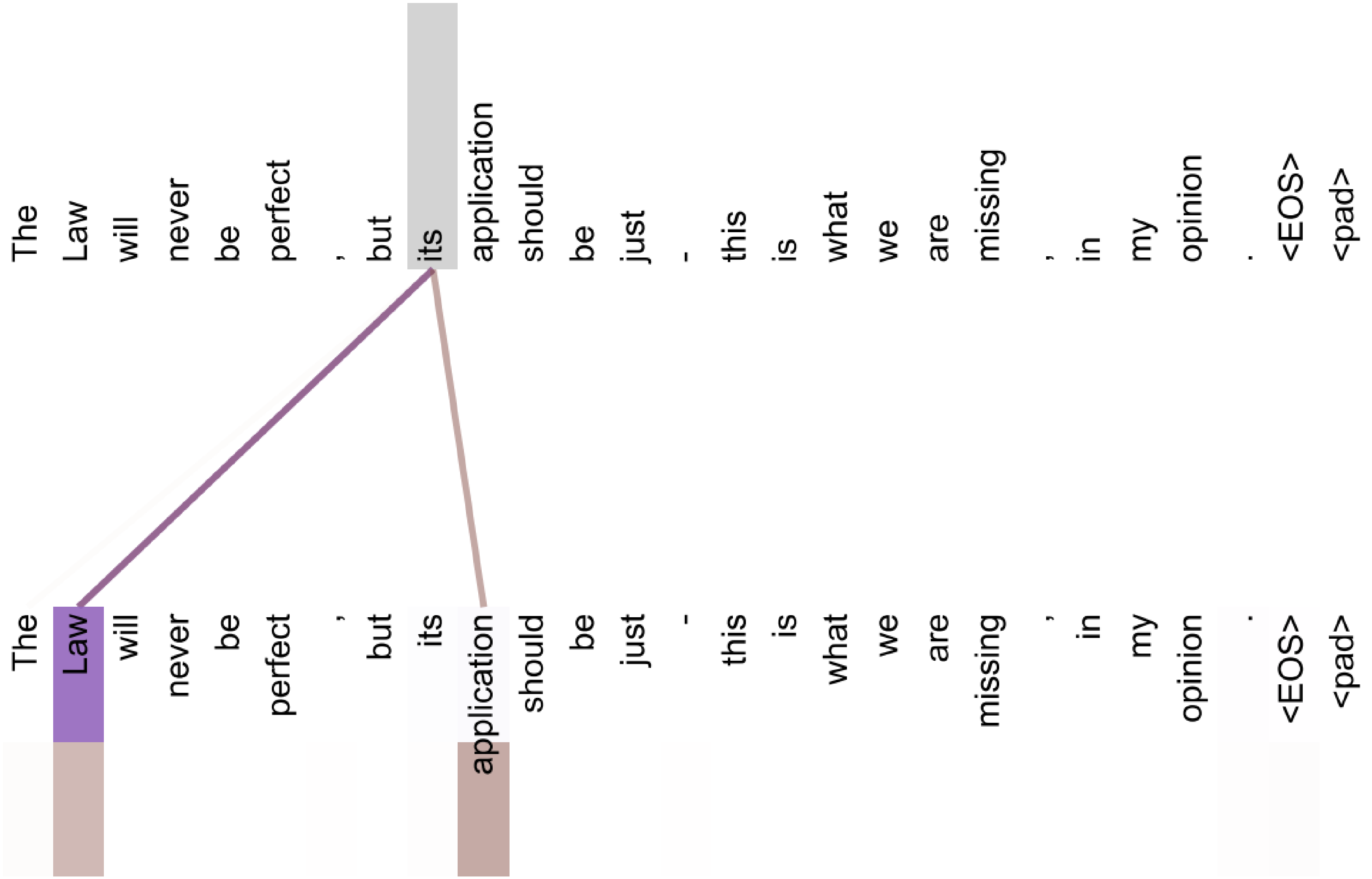
Transformer: Encoder Self-attention Visualization

- Words start to pay attention to other words in sensible ways



Attention Is All You Need, NeurIPS'17

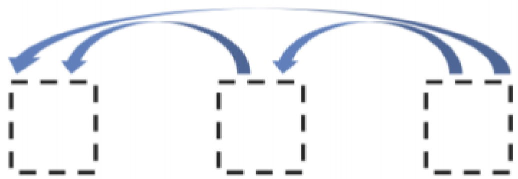
Transformer: Encoder Self-attention Visualization



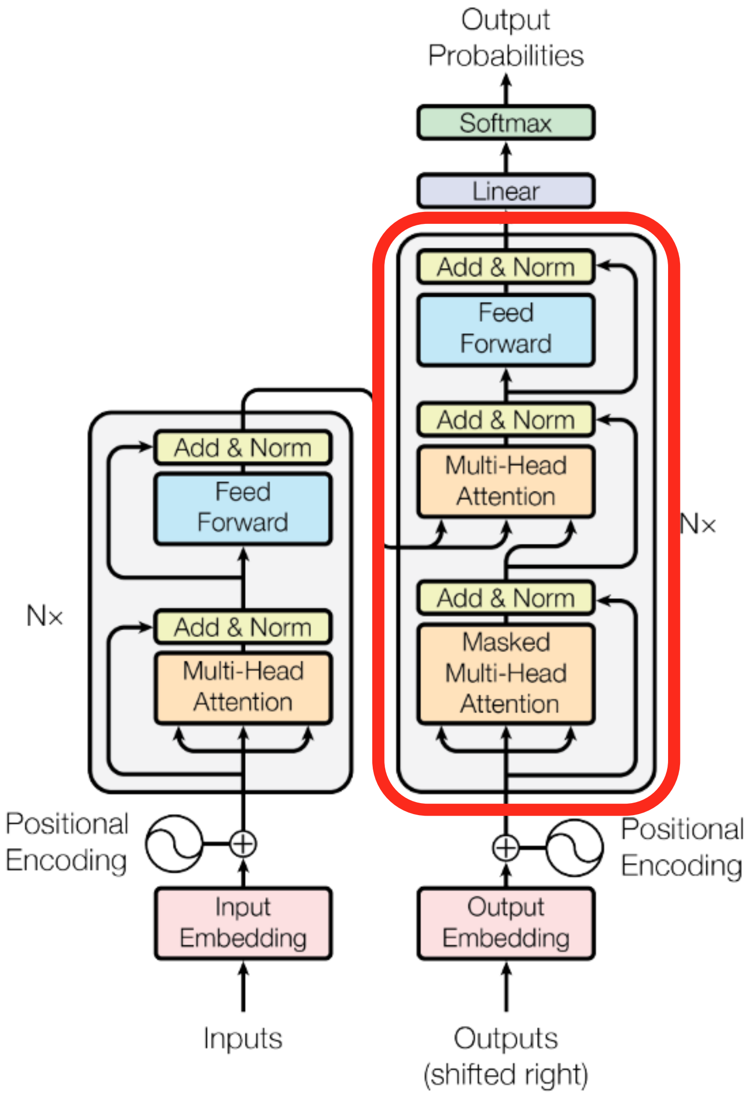
Attention Is All You Need, NeurIPS'17

Transformer: Decoder

- Two sub-layer changes in decoder
- Masked decoder self-attention on previously generated outputs:

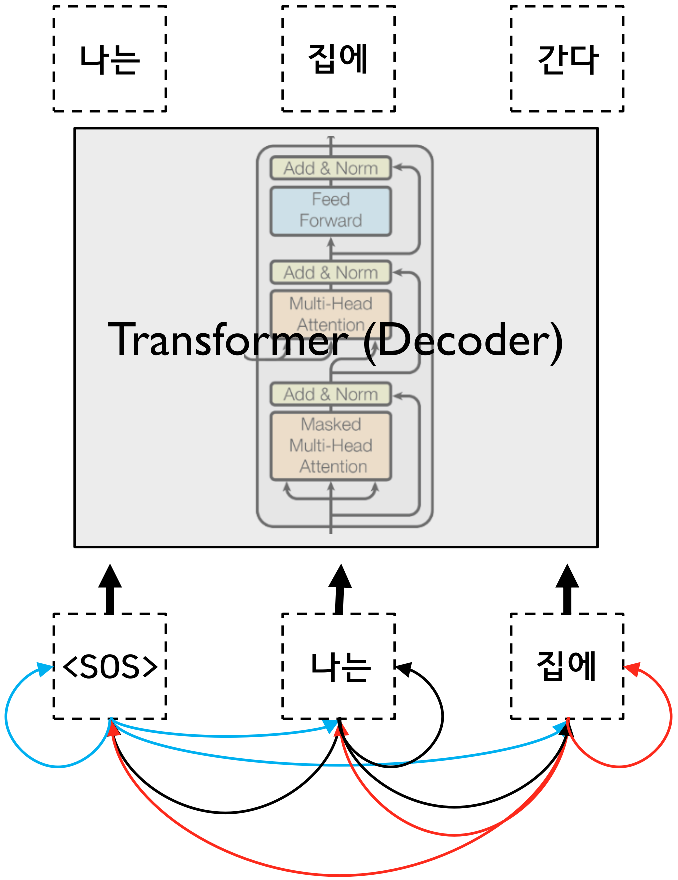


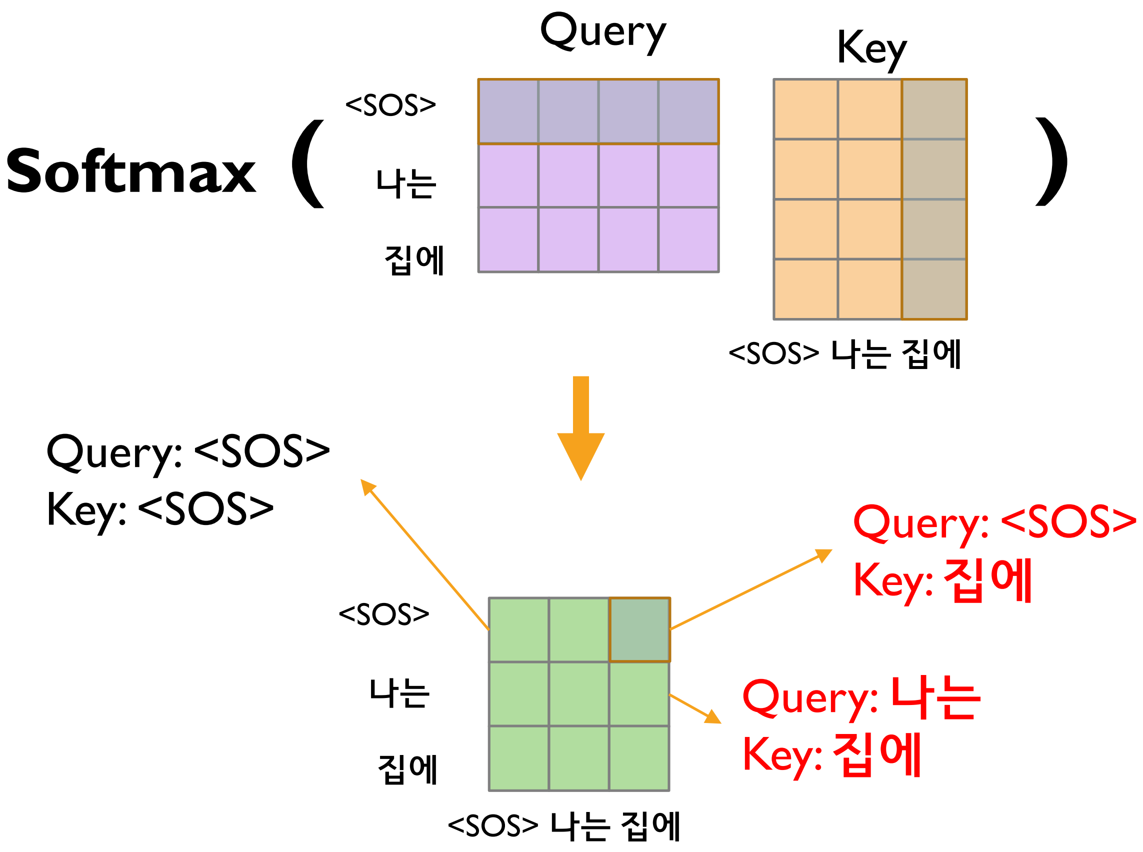
- Encoder-Decoder attention, where queries come from previous decoder layer and keys and values come from output of encoder



Attention Is All You Need, NeurIPS'17

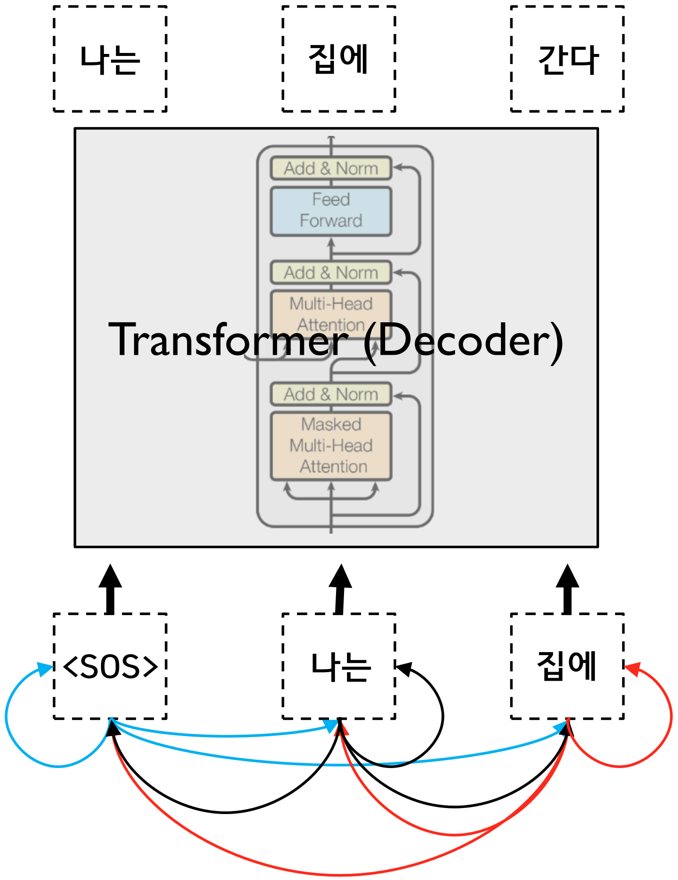
Transformer: Masked Self-Attention

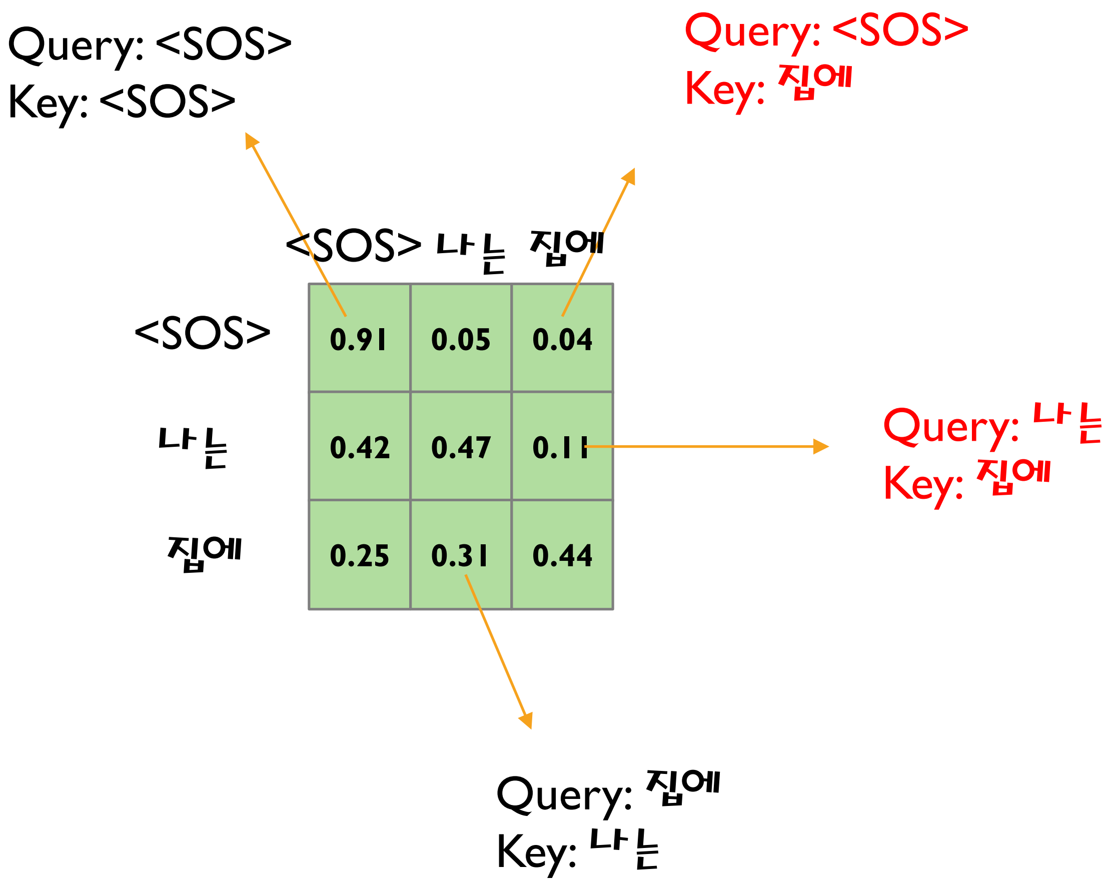




- Those words not yet generated cannot be accessed during the inference time.
- Renormalization of softmax output prevents the model from accessing ungenerated words.

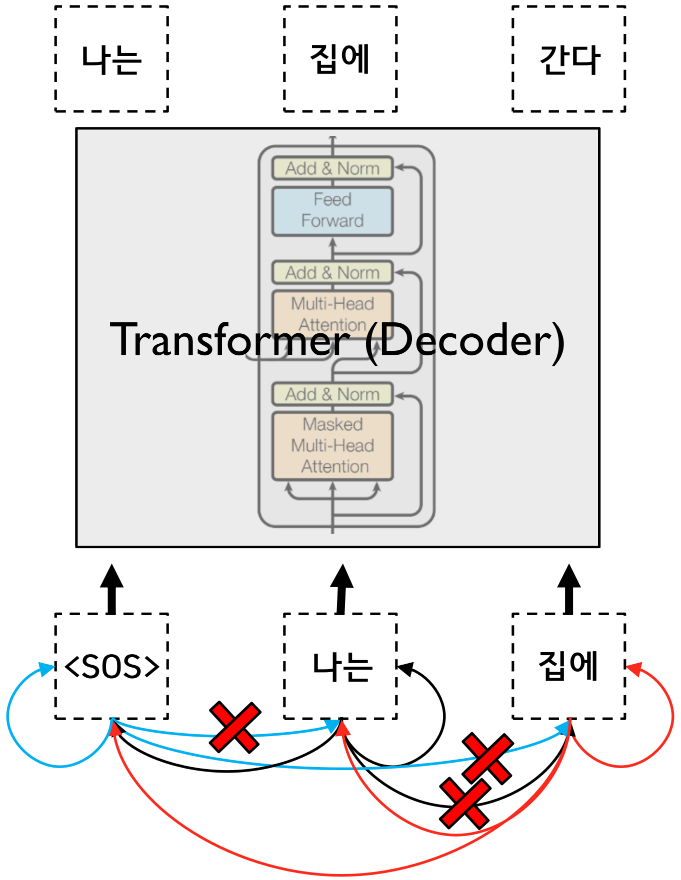
Transformer: Masked Self-Attention

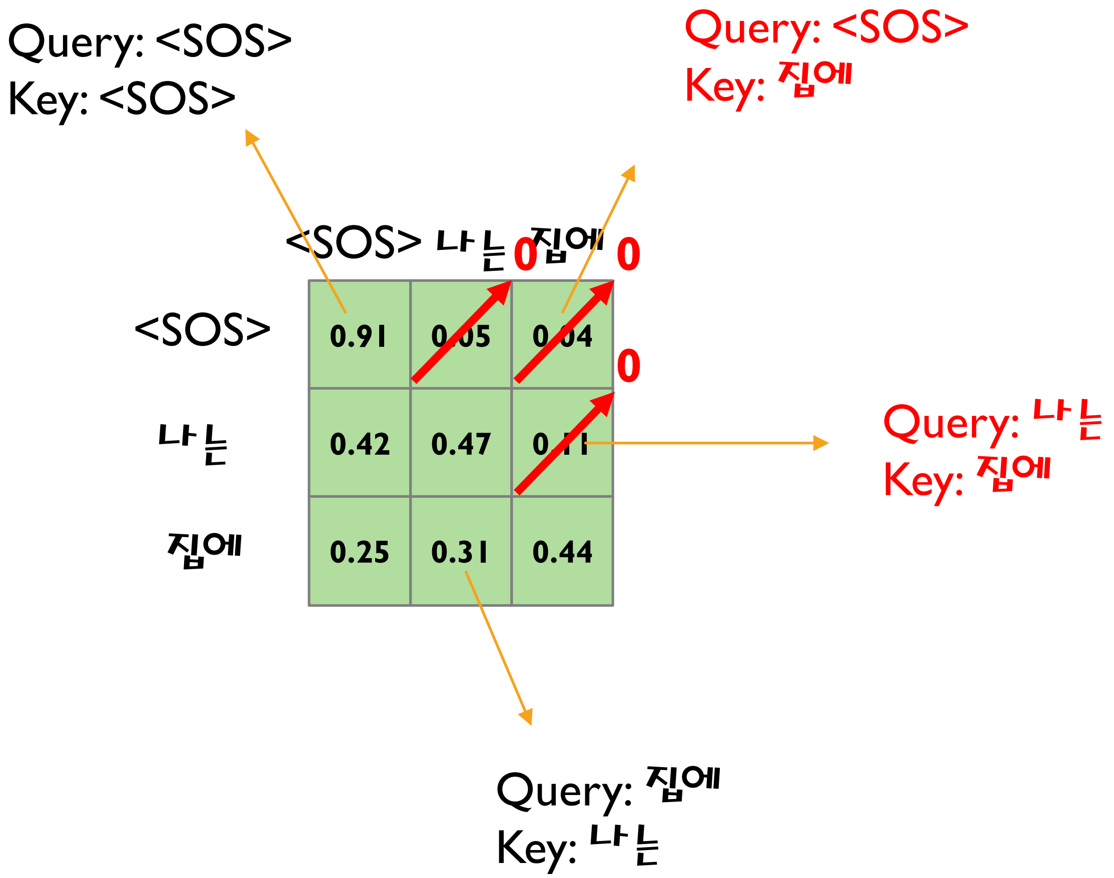




- Those words not yet generated cannot be accessed during the inference time.
- Renormalization of softmax output prevents the model from accessing ungenerated words.

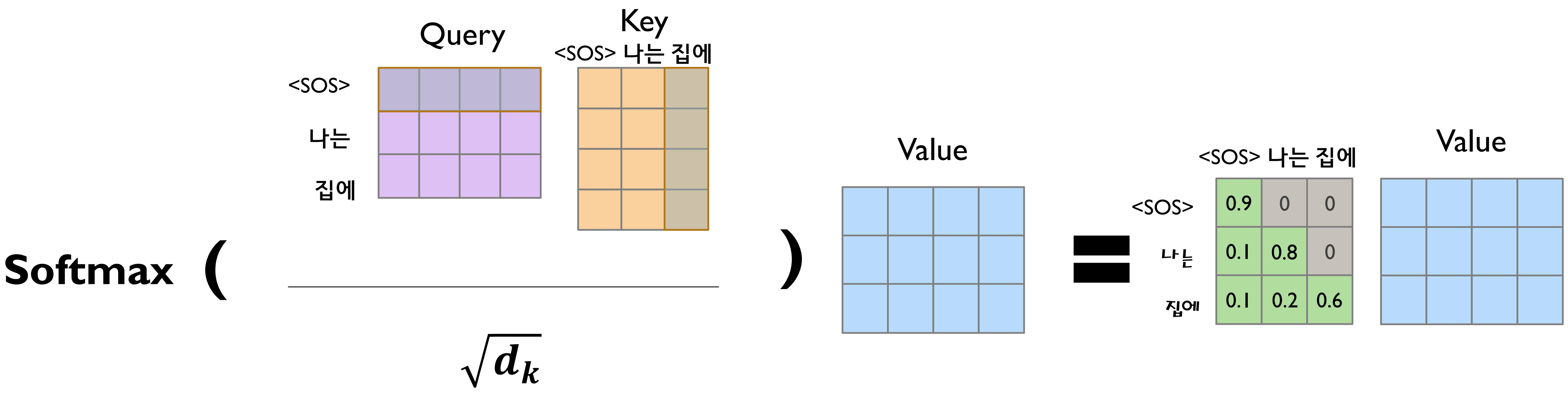
Transformer: Masked Self-Attention





- Those words not yet generated cannot be accessed during the inference time.
- Renormalization of softmax output prevents the model from accessing ungenerated words.

Transformer: Masked Self-Attention



- Those words not yet generated words cannot be accessed during the inference time.
- Renormalization of softmax output prevents the model from accessing ungenerated words.

Attention Is All You Need, NeurIPS'17

Transformer: Experimental Results

Results on English-German/French translation (newstest2014)

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

Recent Trends

- Transformer model and its self-attention block has become a general-purpose sequence (or set) encoder in recent NLP applications as well as in other areas.
- Training deeply stacked Transformer models via a self-supervised learning framework has significantly advanced various NLP tasks through transfer learning, e.g., BERT, GPT-2, XLNet, ALBERT, RoBERTa, Reformer, T5, …
- Other applications are fast adopting the self-attention architecture and self-supervised learning settings, e.g., recommender systems, drug discovery, computer vision, …
- As for natural language generation, self-attention models still requires a greedy decoding of words one at a time.

References

- Harvard NLP - The Annotated Transformer
- Stanford University CS224n – Deep learning for Natural Language Processing
- Fully-parallel text generation for neural machine translation
- Convolution Sequence to Sequence
- The Illustrated Transformer (Eng)
- The Illustrated Transformer (Kor)