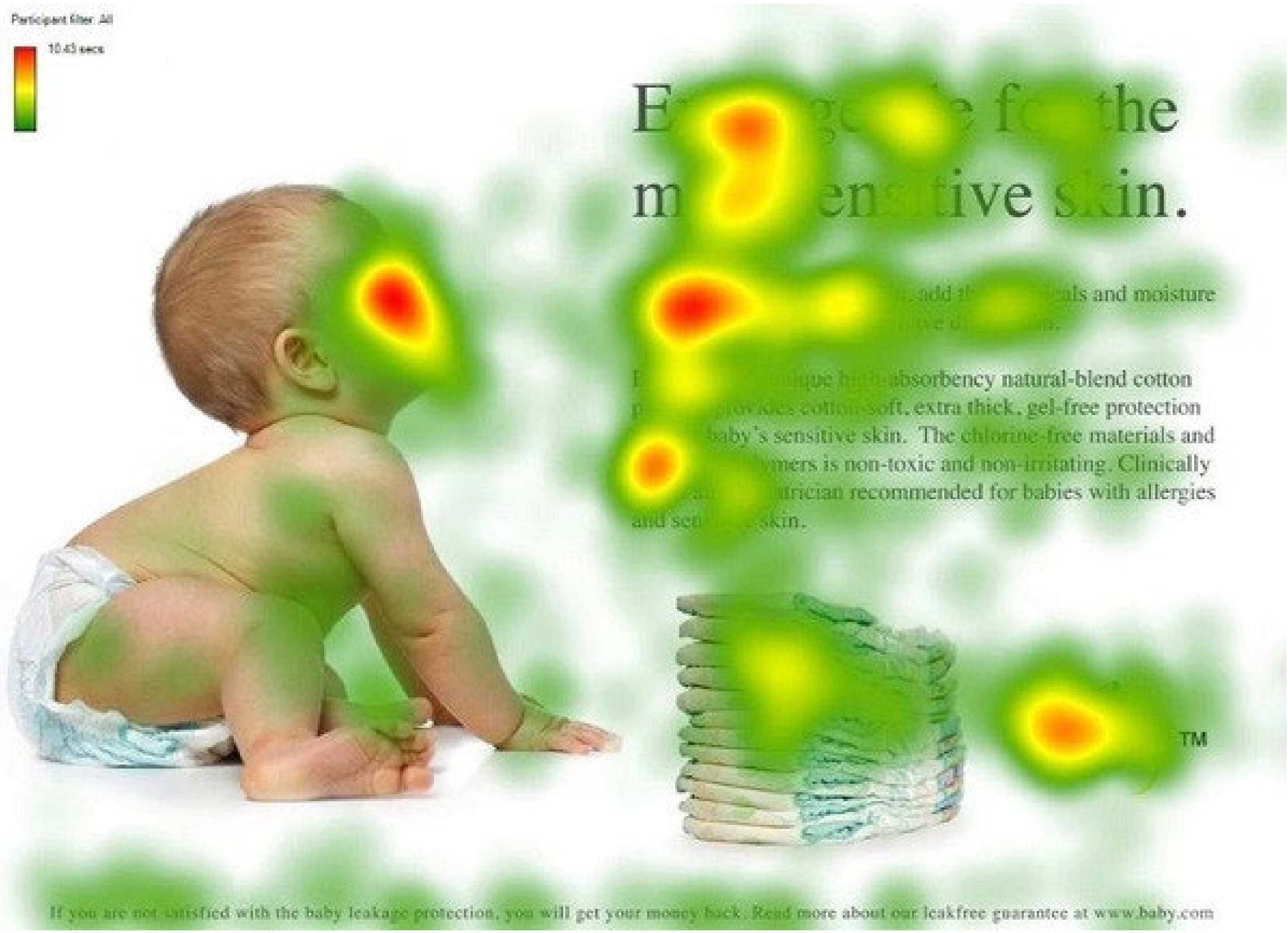
Introduction to Attention

Outline

- What is Attention ?
- Encoder-decoder Attention (seq2seq)
- Self-Attention(Transformer)
- Efficient attention
- Discussion

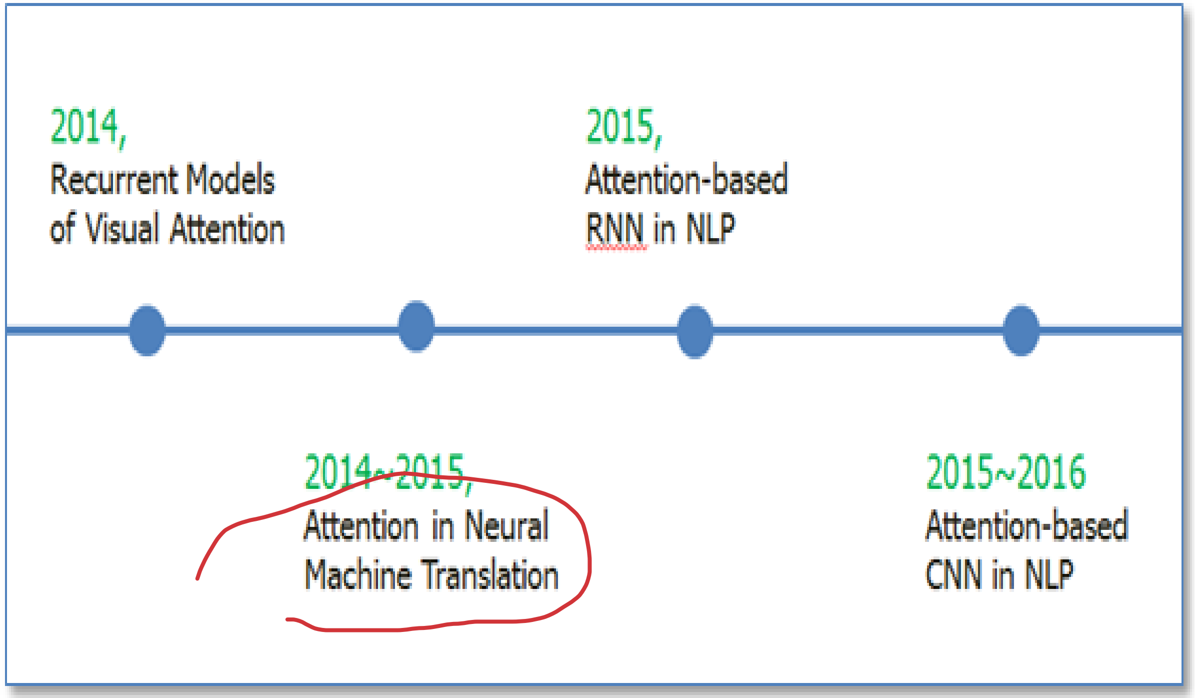


What is Attention?

- **Attention** is simply a weight vector, often the outputs of dense layer using Softmax function.
- **Attention Mechanism** is dynamic weighted sum similar to the human observation mechanism of external things: Selectively to get some important parts of the observed things.

What is Attention?

- It can help model assign different weights to each part of input X, extract more critical and important information, and make update more accurate.
- It explains what the model has learned, provides a window for us to open the black box for deep learning



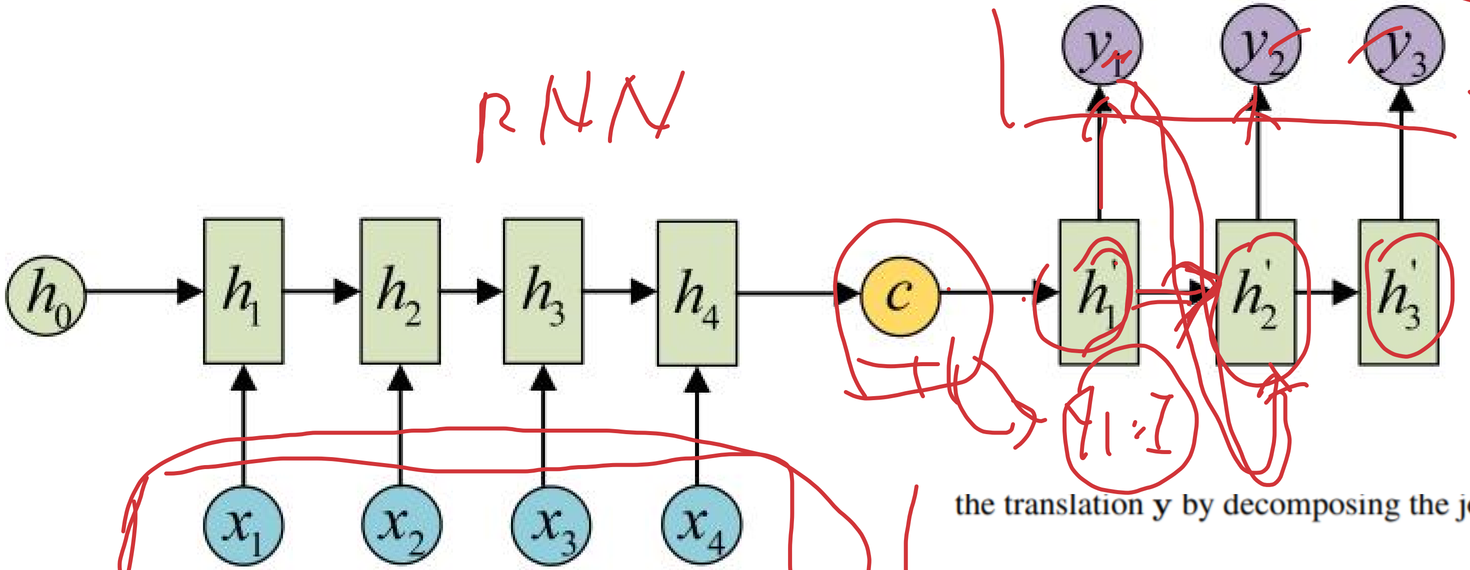
Background/Timeline

- The Attention mechanism was first proposed in the field of Computer Vison around 1990s
- It got popular because of 'Recurrent Models of Visual Attention'
- **'Neural Machine Translation by Jointly Learning to Align and Translate' first apply Attention in NLP(Natural Language Processing) field**
- Attention-based RNN was applied in most aspects of NLP tasks
- Attention-based CNN in NLP
- **'Attention is all you need' is kind of innovation**

Encoder-decoder for seq2seq task

seq2seq

- It is better explained based on one of NLP tasks: Machine Translation.



- All information from X is compressed into fixed length C. Ignore the length of X

the translation **y** by decomposing the joint probability into the ordered conditionals:

$$p(\mathbf{y}) = \prod_{t=1}^{T} p(y_t \mid \{y_1, \cdots, y_{t-1}\}, c), \tag{2}$$

→ X

where $\mathbf{y} = (y_1, \cdots, y_{T_y})$. With an RNN, each conditional probability is modeled as

$$p(y_t \mid \{y_1, \cdots, y_{t-1}\}, c) = g(y_{t-1}, s_t, c), \tag{3}$$

where g is a nonlinear, potentially multi-layered, function that outputs the probability of y_t, and s_t is

RNN:

Encoder-decoder Attention

~~RNN~~ / ~~LSTM~~

2014

- Dynamic context c by weighted sum

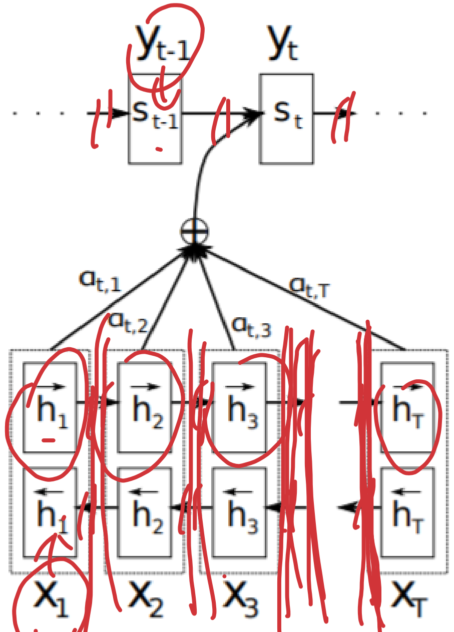


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

$$p(y_i | y_1, \ldots, y_{i-1}, \mathbf{x}) = g(y_{i-1}, s_i, c_i),$$

where s_i is an RNN hidden state for time i, computed by

$$s_i = f(s_{i-1}, y_{i-1}, c_i).$$

The context vector c_i is, then, computed as a weighted sum of these annotations h_i:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j. \quad (5)$$

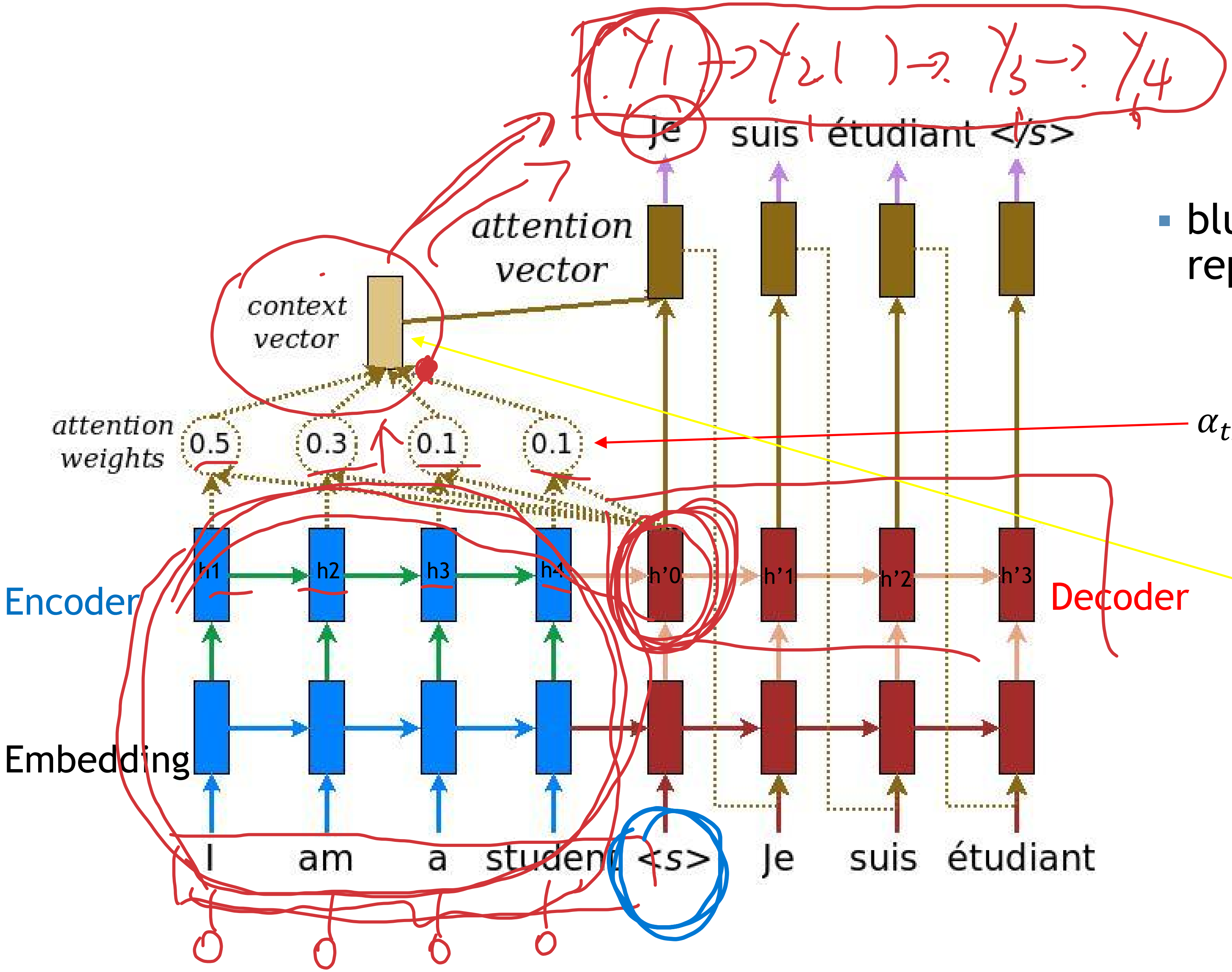
The weight α_{ij} of each annotation h_j is computed by

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})},$$

where

$$e_{ij} = a(\boxed{s_{i-1}}, \boxed{h_j})$$

Hidden state from RNN



- blue represents encoder and red represents decoder

Inner product

$$\alpha_{ts} = \frac{\exp(score(h'_t, h_s))}{\sum_{p=1}^{S} \exp(score(h'_t, h_p))}$$

Q K

$$c_t = \sum_s \alpha_{ts} h_s$$

Attention is all you need (Transformer)

- A paper from Google team in 2017. It was an innovation based on Seq2Seq: Removed RNN. 1
- In common NLP tasks, we split sentence into words or tokens, and transform tokens to word embedding vectors. In that case, each sentence is represented by a matrix(num of word * embedding size)
  - RNN: $y_t = f(y_{t-1}, x_t)$ is the frame of whatever LSTM or GRU. However, RNN cannot learn entire information perfectly, it is nothing but a Markov Decision process.
  - CNN: $y_t = f(x_{t-1}, x_t, x_{t+1})$ is the frame of CNN. It is well used in paper: 'Convolutional Sequence to Sequence Learning' from Facebook. However, CNN can easily parallel and capture entire structure information.
  - Attention: $y_t = f(x_t, A, B)$ is the theme of this paper. More straight-forward in controlling whole structure

Attention is all you need (Transformer)

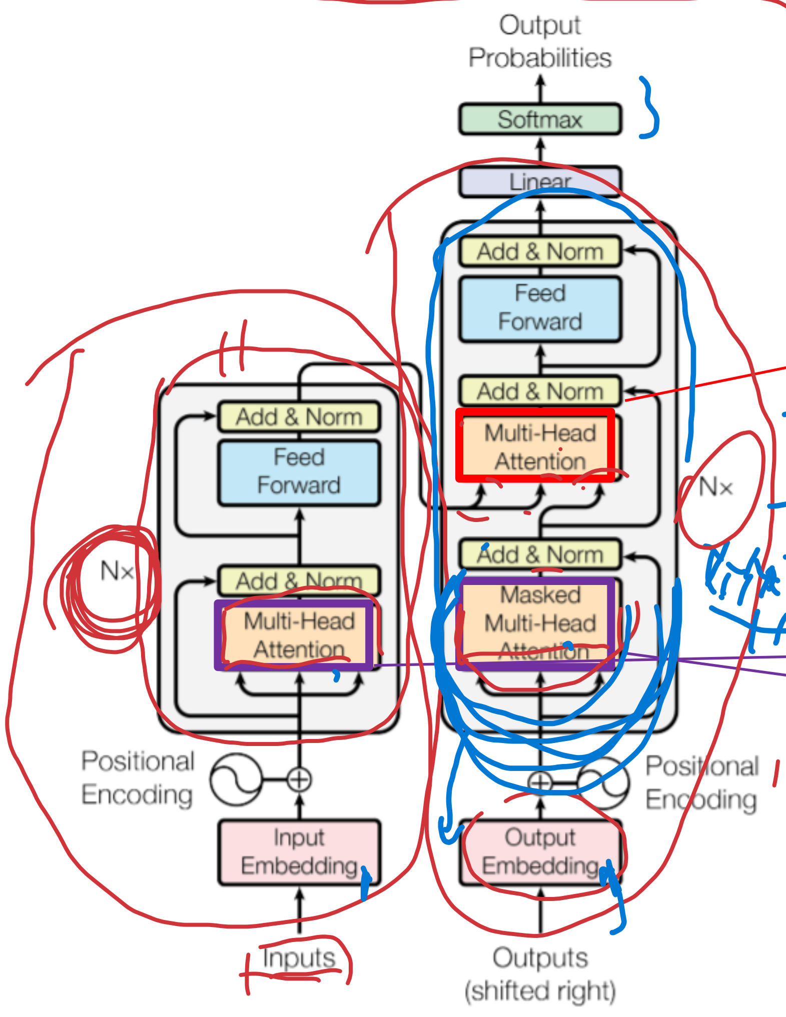


Figure 1: The Transformer - model architecture.

seq - seq

Encoder-decoder attention (no RNN)

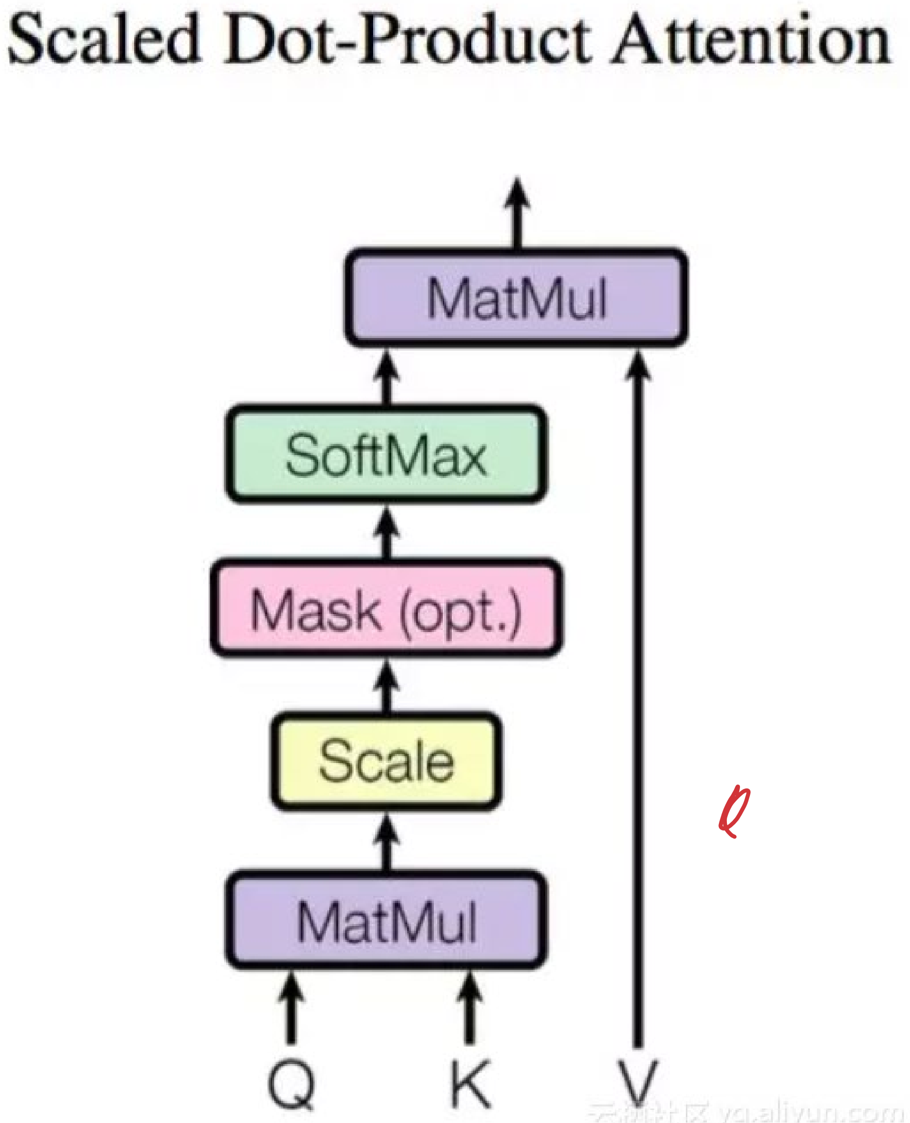
$x_1\ x_2\ x_3\ x_4$

$y_1\ y_2\ y_3\ y_4$

old

new

Self attention:
To **get new embeddings**
based on global infos



Scaled Dot-Product (self Attention)

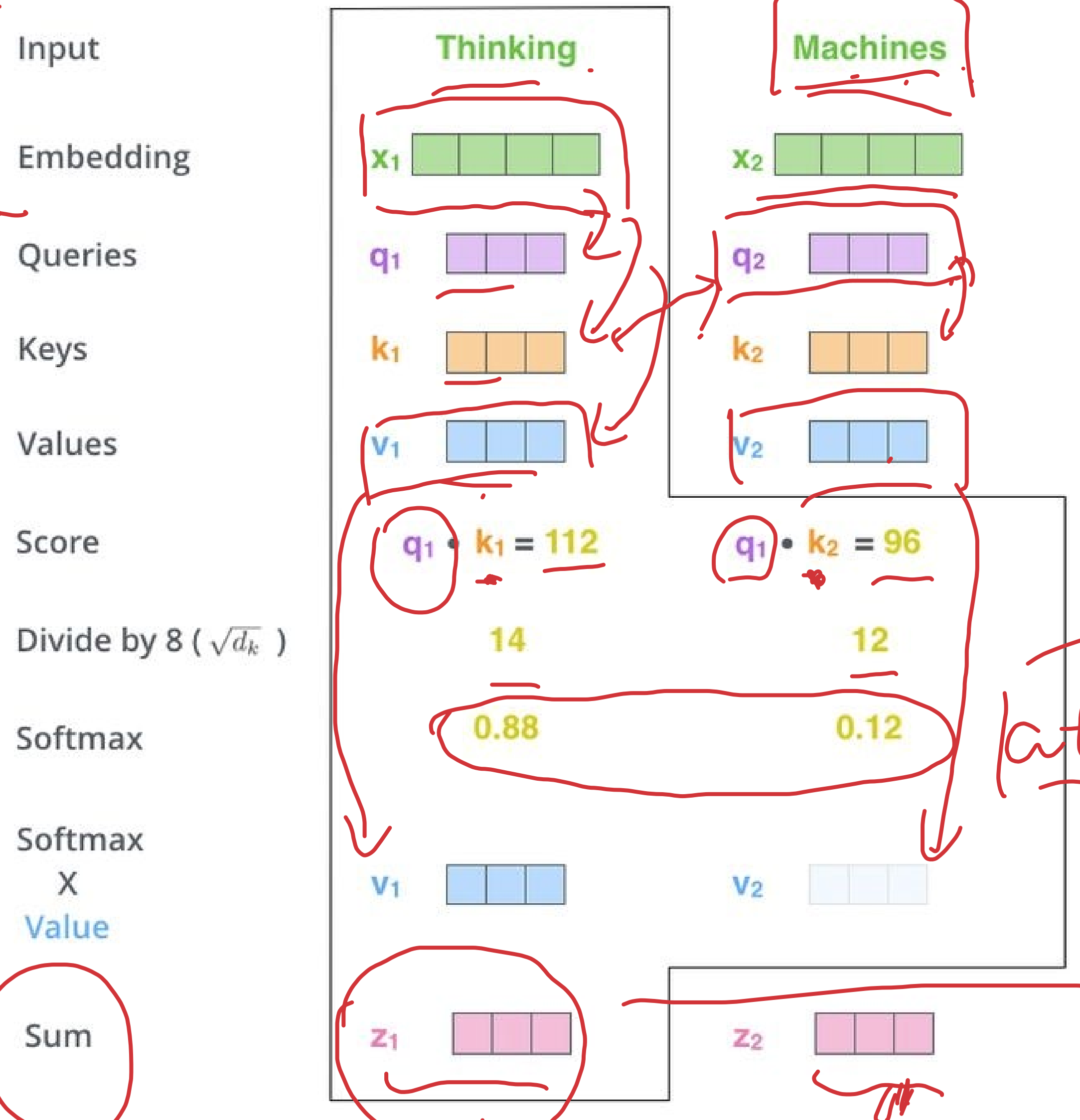
$$Attention(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) = softmax\left(\frac{\boldsymbol{Q}\boldsymbol{K}^{\top}}{\sqrt{d_k}}\right)\boldsymbol{V}$$

$$\boldsymbol{Q} \in \mathbb{R}^{n\times d_k}, \boldsymbol{K} \in \mathbb{R}^{m\times d_k}, \boldsymbol{V} \in \mathbb{R}^{m\times d_v}$$

$$Attention(\boldsymbol{q}_t,\boldsymbol{K},\boldsymbol{V}) = \sum_{s=1}^{m}\frac{1}{Z}\exp\left(\frac{\langle\boldsymbol{q}_t,\boldsymbol{k}_s\rangle}{\sqrt{d_k}}\right)\boldsymbol{v}_s$$

self attention: two words example

input



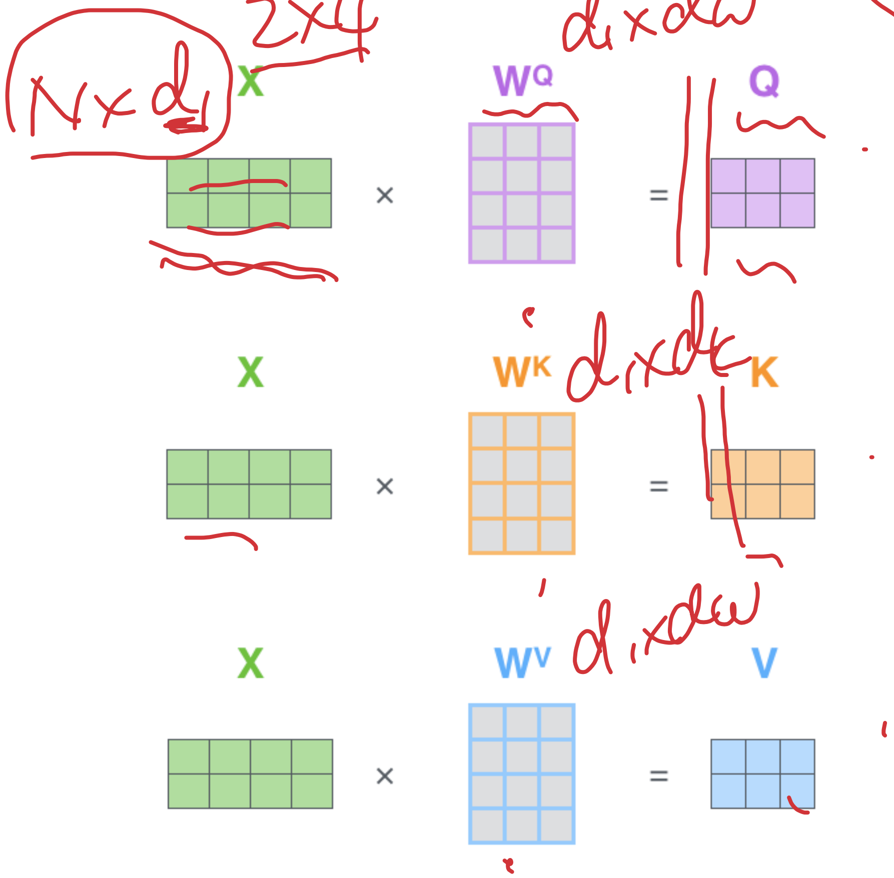
$x = (x_1, x_2)$

atten_vect: weight

$0.88\, v_1 + 0.12\, v_2$

embedding

self attention: two words example matrix-form



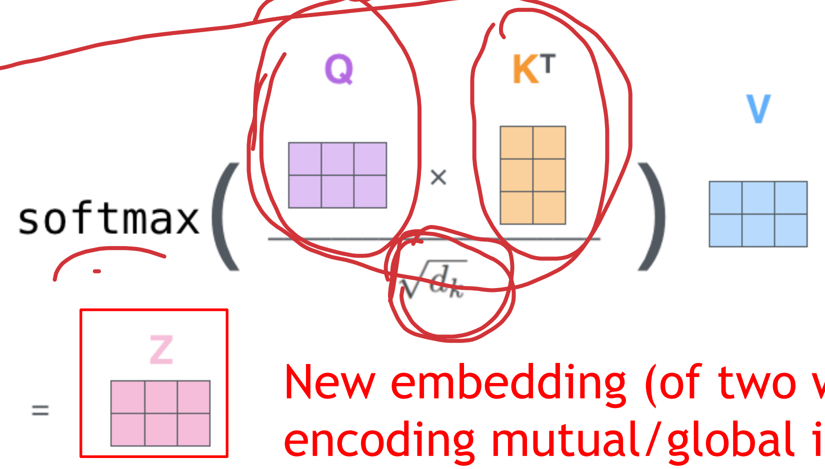
d-atten

N×N

N×dk

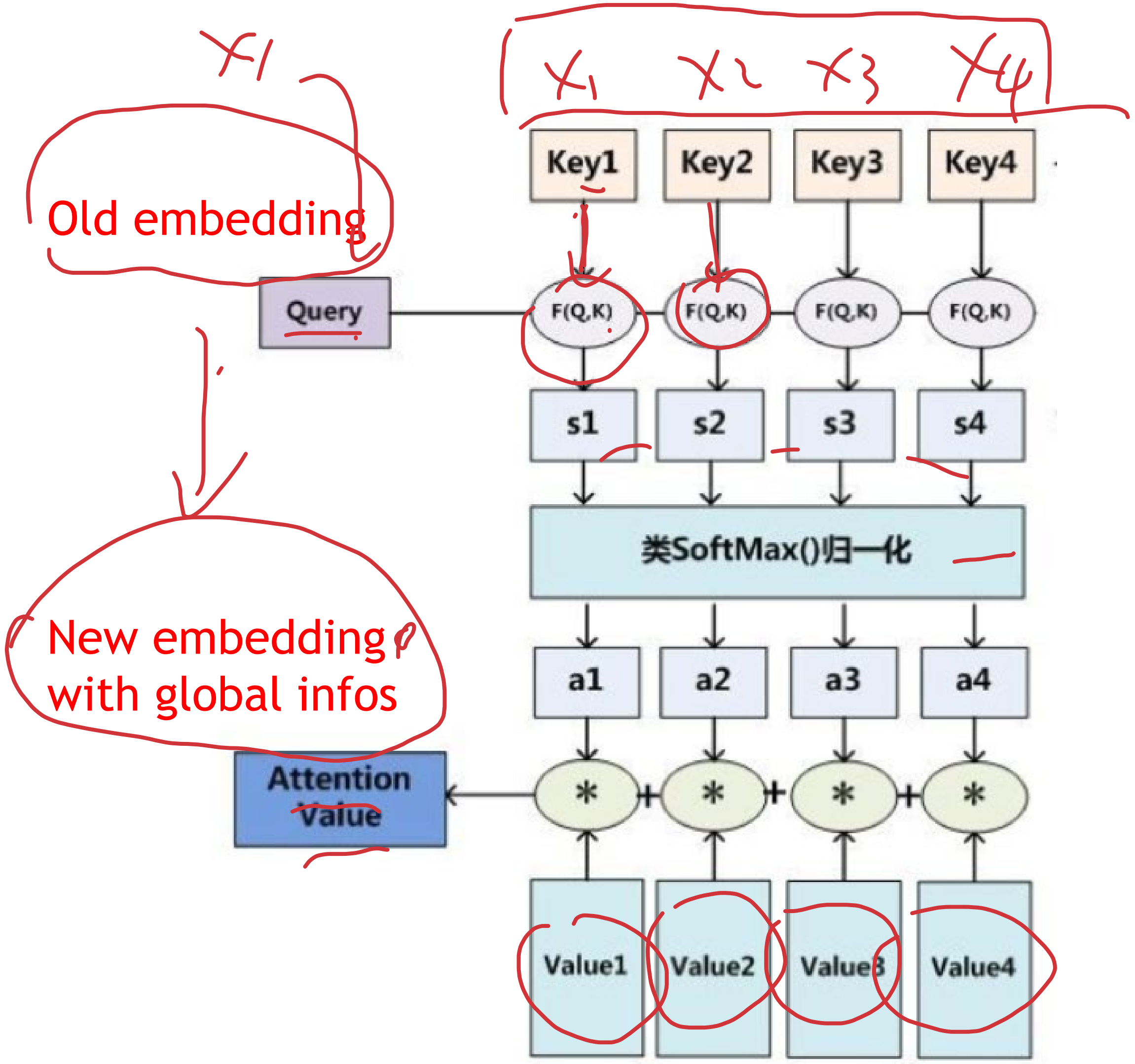
dk×N

O(N²)



New embedding (of two words)!
encoding mutual/global infos

self attention: general form

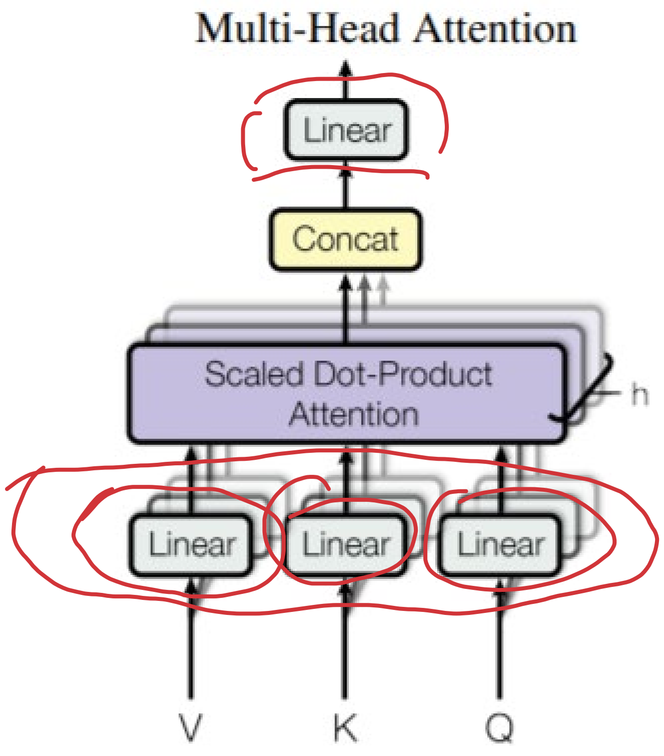


New embedding (of two words)!
encoding mutual/global infos

What are Q, K, V ? Where are they from?

- Q: Query K: Keys V: Values
- Different problems have different Q, K, V
- Usually they are linearly transformed from input or input+output
- In self attention problem, such like figuring out inner structure information of a sentence. You can use Q = K = V = input embedding ,but it is more common to use Q = AX, K = BX, V = CX. A, B and C can be equal

Multi-head : like multi-channel conv kernel in CNN



1x3 Q, K, V 1x5

2x5

$$MultiHead(\mathbf{Q},\mathbf{K},\mathbf{V}) = Concat(head_1,\ldots,head_h)$$

1x100

Q, K, V

1x5

100x10

$$head_i = Attention(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V)$$

1x10

云栖社区 yq.aliyun.com

Attention is all you need (Transformer)

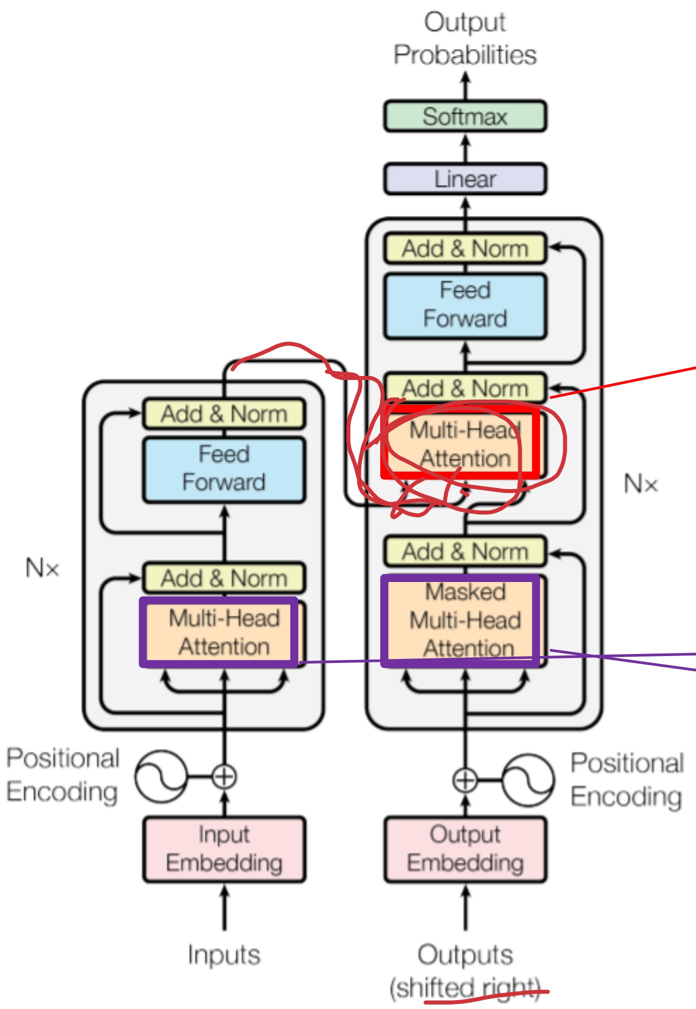


Figure 1: The Transformer - model architecture.

Do it recursively (update the embedding/corresponding Q,K,V)

Encoder-decoder attention (no RNN)

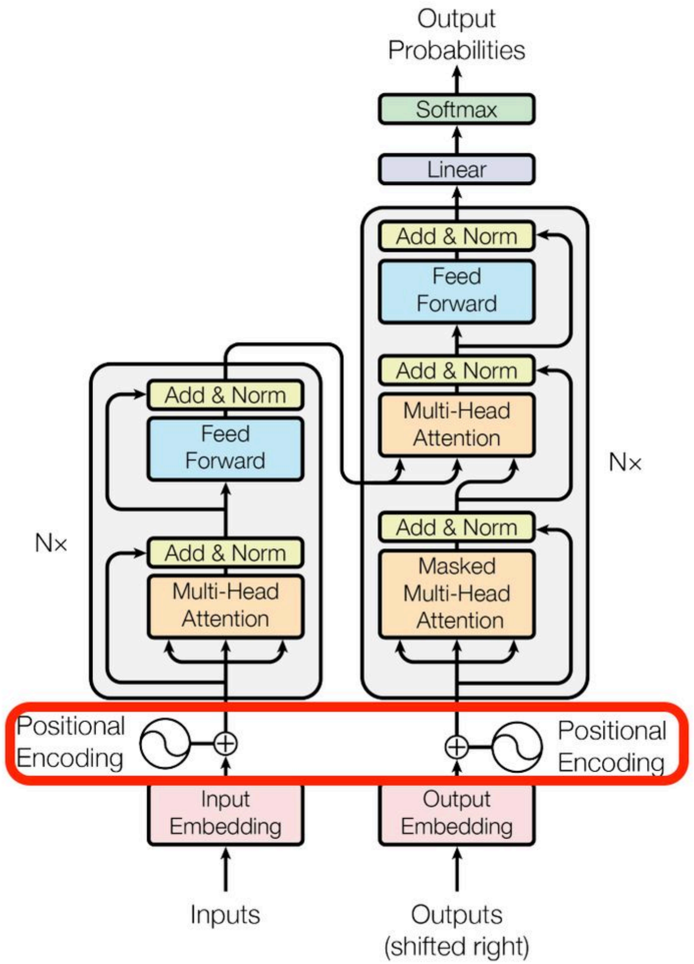
Self attention:

To **get new embeddings** based on global infos

X, Y

Attention is all you need (Transformer)

x1 x2 x3



Self-attention don't encode any position info

Add it manually: sincos / trainable embedding

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

Mask Attention: first self-attention in decoder

Model cannot see the future information

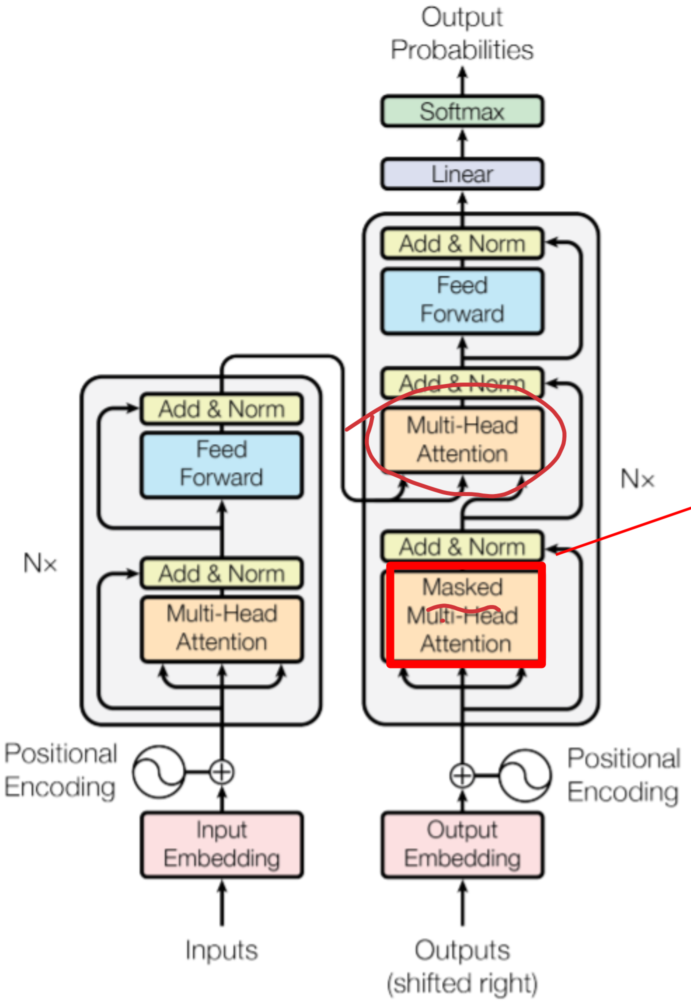


Figure 1: The Transformer - model architecture.

Y_1 Y_2 Y_3 Y_4

	I	have	a	dream
I	1			
have	0.4	0.6		
a	0.1	0.1	0.8	
dream	0.2	0.3	0.1	0.4

知乎 @盛源车

Time limitation of self-attention: T^2

Table 2. Per-layer complexity, minimum number of sequential operations and maximum path lengths for different layer types. T is the sequence length, D is the representation dimension and K is the kernel size of convolutions [137].

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$\boxed{O(T^2 \cdot D)}$	$O(1)$	$O(1)$
Fully Connected	$O(T^2 \cdot D^2)$	$O(1)$	$O(1)$
Convolutional	$O(K \cdot T \cdot D^2)$	$O(1)$	$O(\log_K(T))$
Recurrent	$O(T \cdot D^2)$	$O(T)$	$O(T)$

Position-based Sparse attention

W_K W_Q

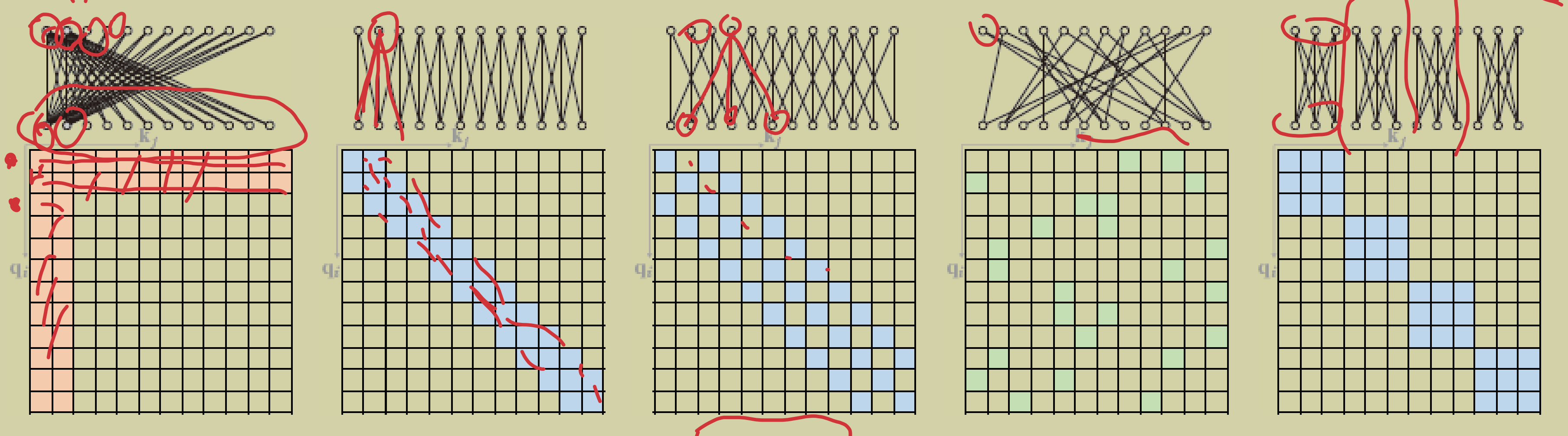


Fig. 4. Some representative atomic sparse attention patterns. The colored squares means corresponding attention scores are calculated and a blank square means the attention score is discarded.

Content-based Sparse attention

Query

A straightforward way of constructing a content-based sparse graph is to select those keys that are likely to have large similarity scores with the given query. To efficiently construct the sparse graph, we can recur to Maximum Inner Product Search (MIPS) problem, where one tries to find the keys with maximum dot product with a query without computing all dot product terms. Routing Transformer [111] uses k-means clustering to cluster both queries $\{\mathbf{q}_i\}_{i=1}^T$ and keys $\{\mathbf{k}_i\}_{i=1}^T$ on the same set of centroid vectors $\{\mu_i\}_{i=1}^k$. Each query only attends to the keys that belong to the same cluster. During training, the cluster centroid vectors are updated using the exponentially moving average of vectors assigned to it, divided by the exponentially moving average of cluster counts:

Rules to update the cluster centroids

$$\tilde{\mu} \leftarrow \lambda\tilde{\mu} + (1-\lambda)\left(\sum_{i:\mu(\mathbf{q}_i)=\mu} \mathbf{q}_i + \sum_{j:\mu(\mathbf{k}_j)=\mu} \mathbf{k}_j\right), \tag{8}$$

$$c_\mu \leftarrow \lambda c_\mu + (1-\lambda)|\mu|, \tag{9}$$

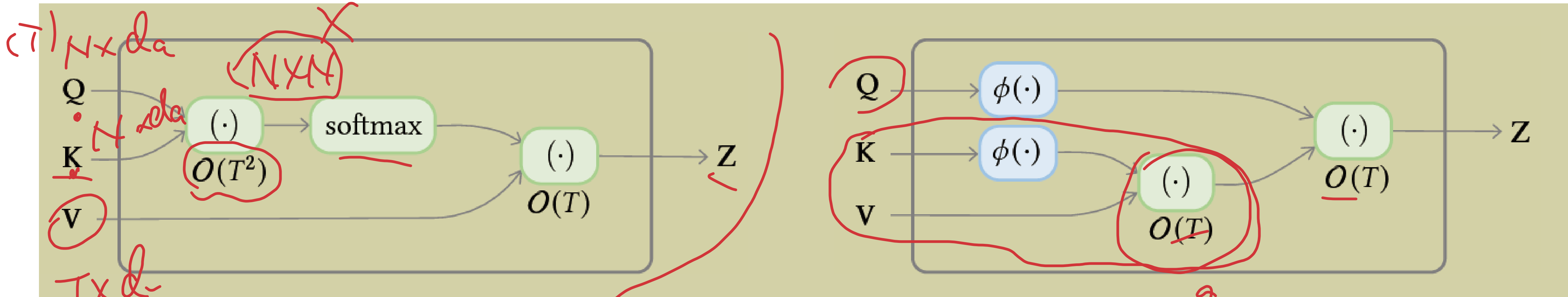
$$\mu \leftarrow \frac{\tilde{\mu}}{c_\mu}, \tag{10}$$

where $|\mu|$ denotes the number of vectors currently in cluster μ and $\lambda \in (0,1)$ is a hyperparameter.

Let $\mathcal{P}_i$ denote the set of indices of keys that the i-th query attend to. $\mathcal{P}_i$ in Routing Transformer is defined as

$$\mathcal{P}_i = \{j : \mu(\mathbf{q}_i) = \mu(\mathbf{k}_j)\}. \tag{11}$$

Linearized Attention



(a) standard self-attention (b) linearized self-attention

$$\mathbf{z}_i = \sum_j \frac{\mathrm{sim}(\mathbf{q}_i, \mathbf{k}_j)}{\sum_{j'} \mathrm{sim}(\mathbf{q}_i, \mathbf{k}_{j'})} \mathbf{v}_j,$$

where $\mathrm{sim}(\cdot,\cdot)$ is a scoring function measuring similarity between input vectors. In vanilla Transformer, the scoring function is the exponential of inner product $\exp(\langle\cdot,\cdot\rangle)$. A natural choice of $\mathrm{sim}(\cdot,\cdot)$ is a kernel function $\mathcal{K}(\mathbf{x},\mathbf{y}) = \phi(\mathbf{x})\phi(\mathbf{y})^\top$, which leads to

$$\mathbf{z}_i = \sum_j \frac{\phi(\mathbf{q}_i)\phi(\mathbf{k}_j)^\top}{\sum_{j'} \phi(\mathbf{q}_i)\phi(\mathbf{k}_{j'})^\top} \mathbf{v}_j \tag{15}$$

1×n n×1 = 1×1

$$= \frac{\phi(\mathbf{q}_i)\sum_j \phi(\mathbf{k}_j) \otimes \mathbf{v}_j}{\phi(\mathbf{q}_i)\sum_{j'} \phi(\mathbf{k}_{j'})^\top}, \tag{16}$$

$$\phi_i(\mathbf{x}) = \mathrm{elu}(x_i) + 1.$$

$$\phi(\mathbf{x}) = \frac{h(\mathbf{x})}{\sqrt{m}} [f_1(\omega_1^\top \mathbf{x}), \cdots, f_m(\omega_m^\top \mathbf{x}), \cdots, f_l(\omega_1^\top \mathbf{x}), \cdots, f_l(\omega_m^\top \mathbf{x})],$$

Low-rank Attention(Nystrom method)

200

Another line of work follow the idea of Nyström method. These Nyström-based methods [16, 152] first select m landmark nodes from the T inputs with down-sampling methods (e.g., strided average pooling). Let $\tilde{Q}, \tilde{K}$ be the selected landmark queries and keys, then the follow approximation is used in the attention computation

$$\tilde{A} = \text{softmax}\left(Q\tilde{K}^\top\right)\left(\text{softmax}\left(\tilde{Q}\tilde{K}^\top\right)\right)^{-1}\text{softmax}\left(\tilde{Q}K^\top\right). \tag{19}$$

T×T

N×N

$T \rightarrow m \rightarrow \tilde{K}$, $\tilde{Q}$

Q

K

Discussion

Attention and Prob model? (kernel / corv matrix)

Non-para attention? (GP?)

References

- [1] Bahdanau, D., Cho, K. & Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. Iclr 2015 1–15 (2014)
- [2] Luong, M. & Manning, C. D. Effective Approaches to Attention-based Neural Machine Translation. 1412–1421 (2015)
- [3] Yin, W., Ebert, S. & Schütze, H. Attention-Based Convolutional Neural Network for Machine Comprehension. 7 (2016)
- [4] Mnih V, Heess N, Graves A. Recurrent models of visual attention[C]//Advances in Neural Information Processing Systems. 2014: 2204-2212.
- [5] Yang, Z. et al. Hierarchical Attention Networks for Document Classification. Naacl (2016)
- [6] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., & Gomez, A. et al. . *Attention Is All You Need*. *arXiv: 1706.03762* (2017)
- [7] Mullenbach, James, et al. "Explainable Prediction of Medical Codes from Clinical Text." *arXiv preprint arXiv:1802.05695*(2018).
- [8] Gehring, Jonas, et al. "Convolutional sequence to sequence learning." *arXiv preprint arXiv:1705.03122* (2017).
- [9] Hermann, K. M. et al. Teaching Machines to Read and Comprehend. arXiv 1–13 (2015)