Attention & Self-Attention

Zhongsheng Wang
University of Auckland

Mar 15, 2024

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND

Paper link: https://arxiv.org/pdf/1706.03762.pdf

Table of Contents

Problems with RNN/LSTM

- Memory length is limited (RNN) Long-term memory
- Forgot key information (LSTM)
- Cannot be parallelized (Calculate t_0, then t_1, t_2 …)
  - Training efficiency is relatively low

Attention

- What do you see at first glance in the picture on the left?
- **Human:** Limited attention span, focusing on one thing in a short period (finding the highest priority thing to do among the mess of things)
- **Computer:** Limited computing resources, and we hope it can efficiently capture critical information from a large amount of information.

Attention is all you need

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

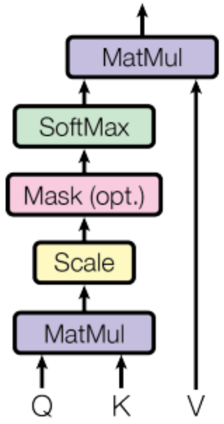
Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

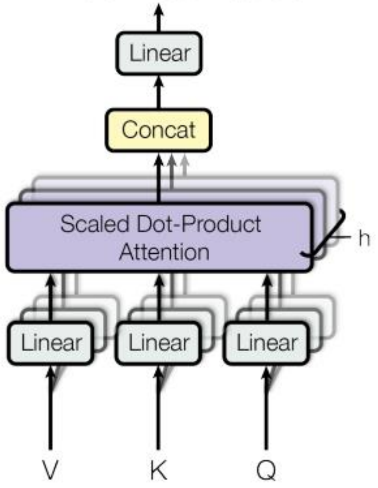
Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Scaled Dot-Product Attention



Multi-Head Attention



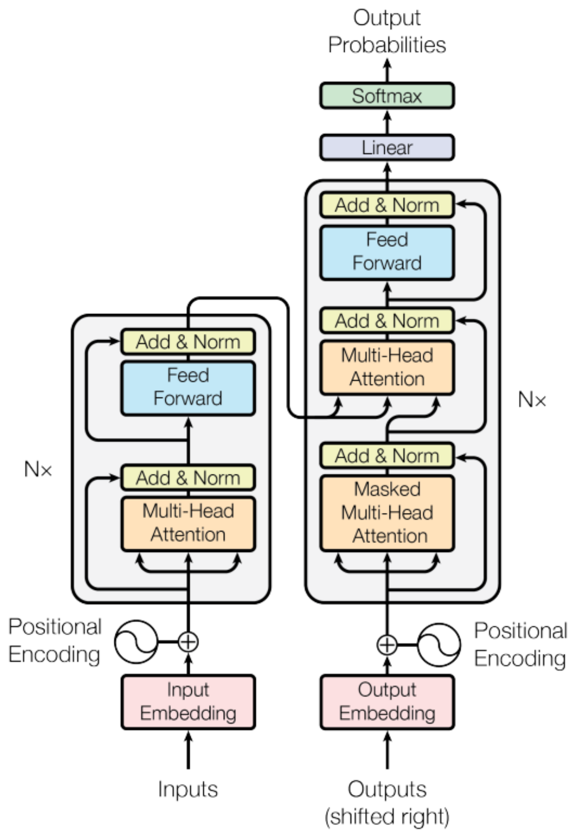


Figure 1: The Transformer - model architecture.

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

$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

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

Paper link: https://arxiv.org/pdf/1706.03762.pdf

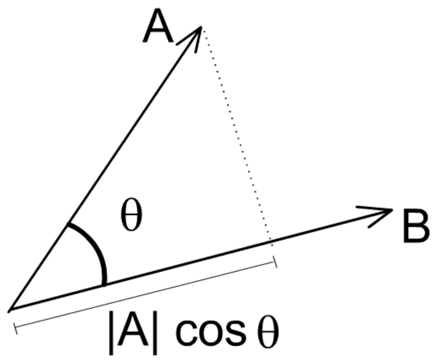
Self-attention

- **Input**: A sentence with several words. **Output**: Word vector for each word.
- Create Matrix QKV from original input x
- Q: Query
  - Will match each k
- K: Key
  - Will be matched by every q
- V: Value
  - Extract value Information from x

- Judge the similarity using QK and rewrite the representation of a word vector using V (match the most similar result)

Better understanding of Softmax(XX^T)X

- Dot-product: Similarity between 2 word vectors
- Word vec1: $[1, 3, 2]^T$ Word vec2: $[4, 1, 1]^T$ (Column vector)
- Dot-product = 1*4 + 3*1 + 2*1 = 9
- Self dot-product for vec1 = 1*1 + 3*3 + 2*2 = 14
- In matrix, dot-product => vec * vec^T (Matrix transpose) = $a_1*a_1 + a_2*a_2 + \ldots$



Cont.

- If we combine all word vectors:
- $X = [x_1, x_2, x_3] = \begin{bmatrix} x_{11} & x_{21} & x_{31} \\ x_{12} & x_{22} & x_{32} \\ x_{13} & x_{23} & x_{33} \end{bmatrix}$ (word vector dimension is 3)
- $XX^T = [x_1, x_2, x_3] \begin{bmatrix} x1 \\ x2 \\ x3 \end{bmatrix} = \begin{bmatrix} x_{11} & x_{21} & x_{31} \\ x_{12} & x_{22} & x_{32} \\ x_{13} & x_{23} & x_{33} \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix}$
- Calculate the similarity between two pairs (including themselves)

Cont.

$$\mathrm{Softmax}(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

- Softmax(XX^T)X: convert the dot-product weight value back to the presentation of an average word vector
- Softmax(XX^T): weight value   X: word vec representation
- So the result of <u>Softmax(XX^T)X</u> is a new word vector representation that is weighted and summed by the attention mechanism

Scaled Dot-Product Attention

$$\text{Softmax}(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

Key

$K = W_k X$

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

Query

$Q = W_q X$

Value

$V = W_v X$

Dimension of K

$$Q = W_q X$$
$$K = W_k X$$
$$V = W_v X$$
$$A = K^T Q$$
$$A' = softmax(A)$$
$$Y = VA'$$

Size of each variable

$$Q = W_q X$$
$$K = W_k X$$
$$V = W_v X$$
$$A = K^T Q$$
$$A' = softmax(A)$$
$$Y = VA'$$

- Size of X: (a, b) **column vector**
  - a: the dimension of a word vector
  - b: number of words in a corpus(a sentence)
- Size of W_{qkv}: (c, a) => Size of QKV: (c, b)
  - $W = QX^T$ (Random Start)
- Size of A: (b, b) = (b, c)*(c, b)

Cont.

$$Q = W_q X$$
$$K = W_k X$$
$$V = W_v X$$
$$A = K^T Q$$
$$A' = softmax(A)$$
$$Y = VA'$$

- A: [a_1, a_2, a_3, …, a_b] a_1: [a_{11}, a_{12},…,a_{1b}]
- Softmax for each column
  - A: [a'_1, a'_2, a'_3, …, a'_b]
  - a_1: [a'_{11}, a'_{12},…,a'_{1b}] (b: number of words in a sentence)
- Y: (c, b) => (c, b)*(b, b)
- So the output is still the representation of each word
- c is a hyperparameter (set whatever u want)

Calculation demo

- Suppose:
  - 4 words in a sentence (x1, x2, x3), dimension of the word vector is 4
  - Size of X: (a, b) => a=4(dimension) b=3(num of words)
    - For each x in X: size of x: (a, 1) => (4, 1)
  - $x1 = [1, 2, 3, 4]^T$
  - $x2 = [3, 2, 1, 3]^T$
  - $x3 = [0, 1, 1, 4]^T$
  - All of them are column vectors

Cont.

- Random start Wq Wk Wv: (Size of W: (c, a) suppose c is also 4) => (4, 4) so d_k is also 4

- $Wq = \begin{bmatrix} 3 & 1 & 1 & 3 \\ 3 & 5 & 0 & 0 \\ 4 & 7 & 1 & 1 \\ 2 & 1 & 2 & 3 \end{bmatrix}$ $Wk = \begin{bmatrix} 2 & 2 & 0 & 1 \\ 4 & 2 & 3 & 1 \\ 0 & 2 & 4 & 1 \\ 1 & 4 & 2 & 1 \end{bmatrix}$ $Wv = \begin{bmatrix} 1 & 3 & 2 & 4 \\ 2 & 0 & 1 & 1 \\ 3 & 1 & 3 & 4 \\ 2 & 1 & 1 & 0 \end{bmatrix}$

Cont.

- For x1, calculate q1, k1, v1:
- q1 = Wx*x1 = $\begin{bmatrix} 2 & 2 & 0 & 1 \\ 4 & 2 & 3 & 1 \\ 0 & 2 & 4 & 1 \\ 1 & 4 & 2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 2*1+2*2+1*4 \\ 4*1+2*2+3*3+1*4 \\ 2*2+3*4+1*4 \\ 1*1+4*2+2*3+1*4 \end{bmatrix} = \begin{bmatrix} 10 \\ 18 \\ 15 \\ 19 \end{bmatrix}$
- Q = [q1, q2, q3, q4] = Wx*[x1, x2, x3, x4] =Wx*X (size:(4, 4))
- Each col represents a query vector of a word
- Same for x2, x3 and x4

Cont.

- To calculate the similarity of x1 and x2: q1= $\begin{bmatrix} 10 \\ 18 \\ 15 \\ 19 \end{bmatrix}$
  - Use q1*k2 and then Softmax

  - k2 = Wk*x2 = $\begin{bmatrix} 2 & 2 & 0 & 1 \\ 4 & 2 & 3 & 1 \\ 0 & 2 & 4 & 1 \\ 1 & 4 & 2 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \\ 1 \\ 3 \end{bmatrix} = \begin{bmatrix} 2*3+2*2+1*3 \\ 4*3+2*2+1*3+1*3 \\ 2*2+1*4+1*3 \\ 1*3+4*2+2*1+1*3 \end{bmatrix} = \begin{bmatrix} 13 \\ 22 \\ 11 \\ 16 \end{bmatrix}$

  - So α_{12} = $\begin{bmatrix} 10 \\ 18 \\ 15 \\ 19 \end{bmatrix} * \begin{bmatrix} 13 \\ 22 \\ 11 \\ 16 \end{bmatrix}$ = 10*13+18*22+15*11+19*16 = 995

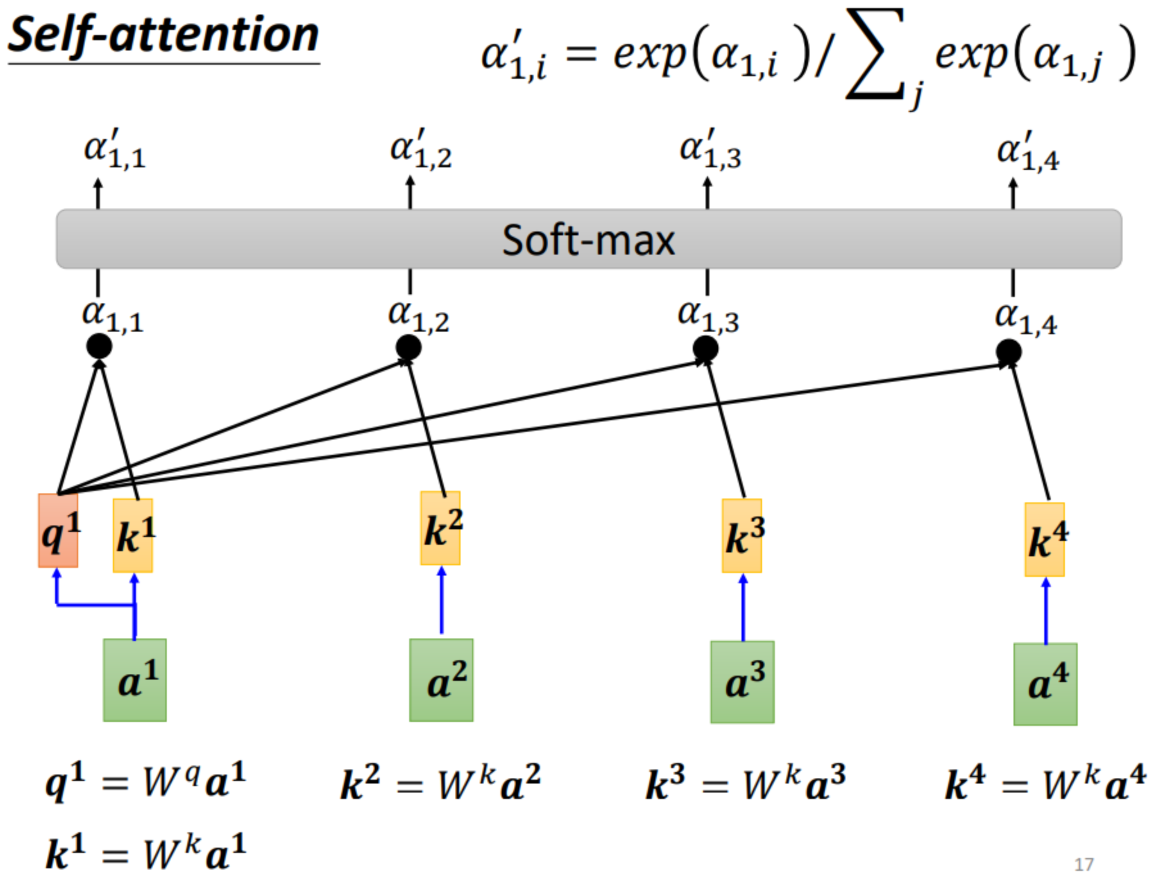
Cont.

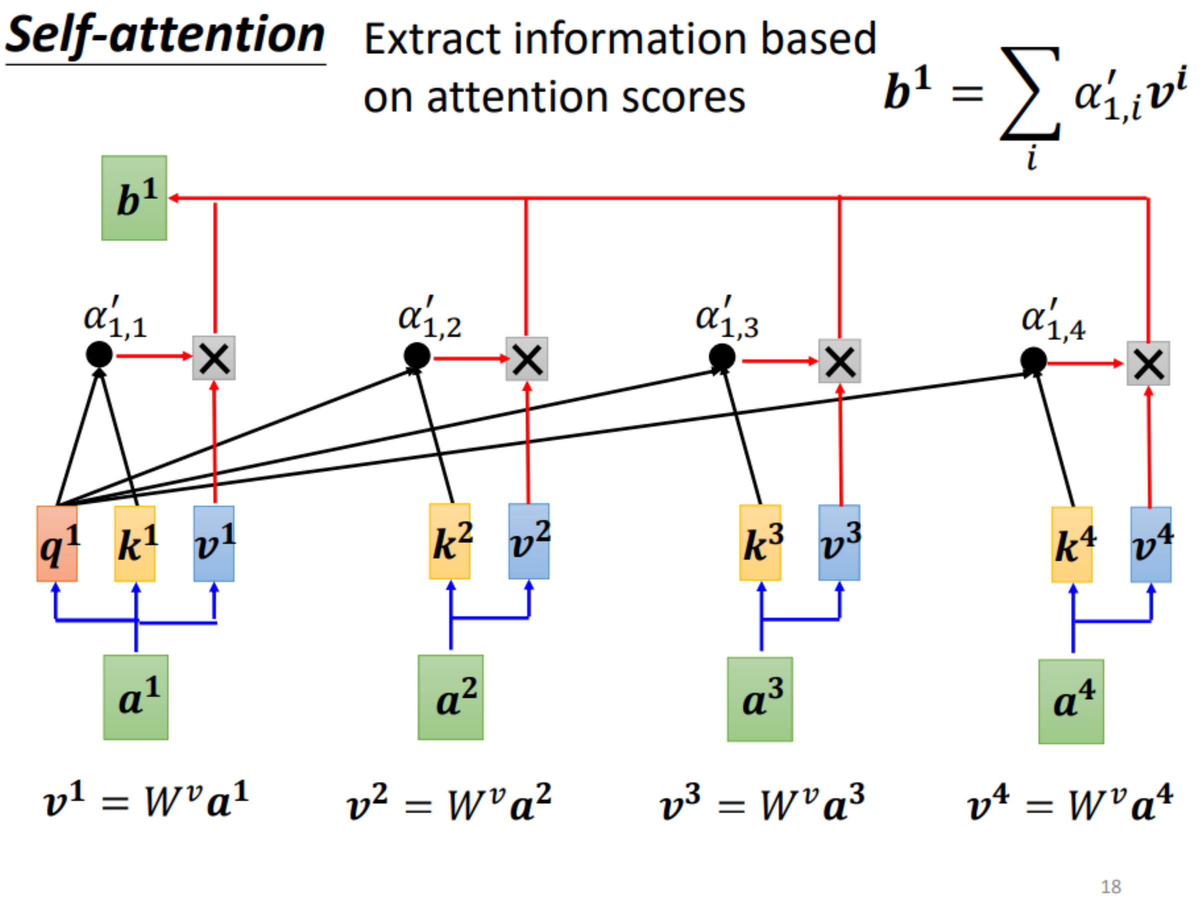
- Softmax after calculating all the similarity of x1 with others (contain itself) ($\alpha_{11}\,\alpha_{12}\,\alpha_{13}\,\alpha_{14}$)
- $\alpha_{11} = \alpha_{11} \,/\, \sqrt{d_k}$ (according to the formula)
  - Value after dot-product becomes very large, and the gradient disappears after softmax
- $\alpha'_{11} = \dfrac{e^{\alpha_{11}}}{e^{\alpha_{11}}+e^{\alpha_{12}}+e^{\alpha_{13}}+e^{\alpha_{14}}}$ (Softmax)
- Same way to calculate $\alpha'_{11}\,\alpha'_{12}\;\alpha'_{13}\,\alpha'_{14}$
- y1 = [v1, v2, v3, v4]*[α'_{11}, α'_{12},α'_{13},α'_{14}]^T

Cont.

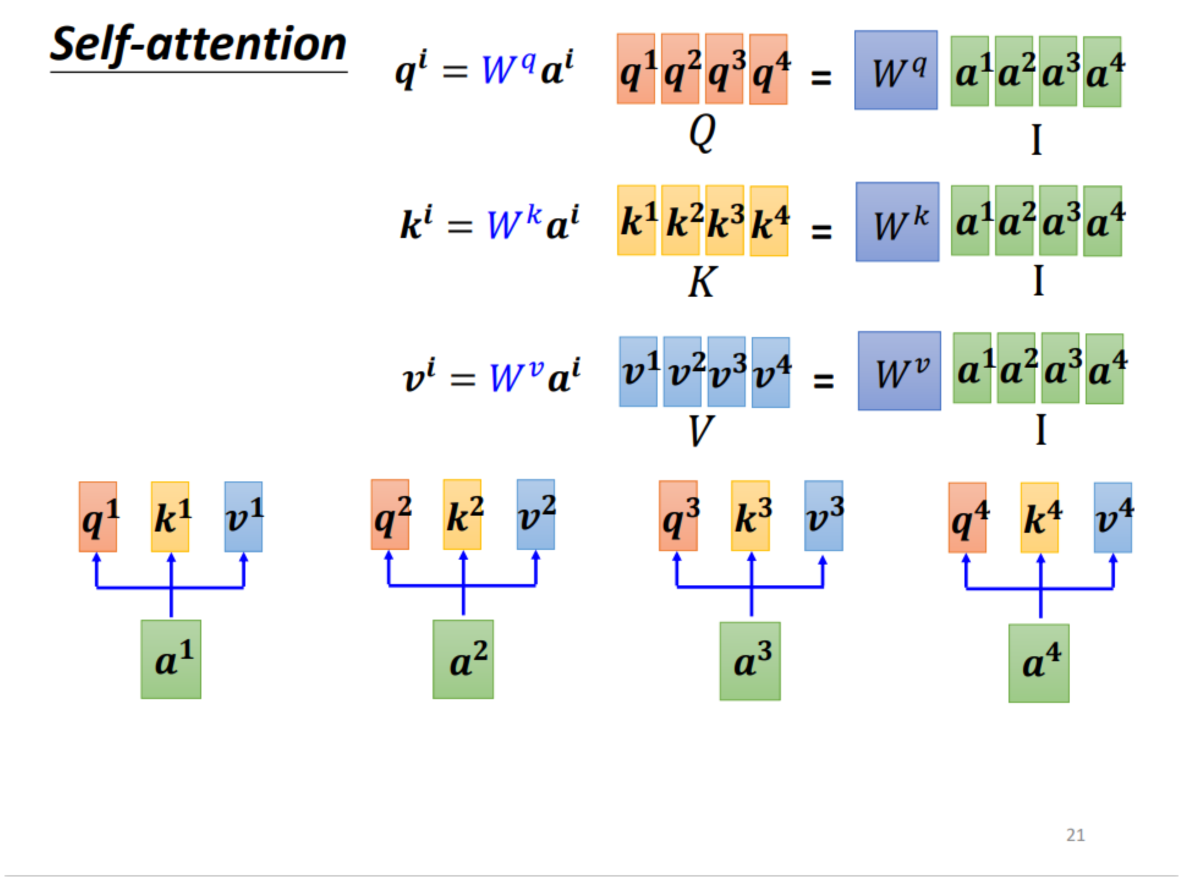
- y1 = [v1, v2, v3, v4]*[α'_{11}, α'_{12},α'_{13},α'_{14}]^T
- Size of v1: (4, 1) V: (4, 4) y1: (4, 1)*(4, 1)[dot-product] = (4, 1)
- So y1 has the same size as x1, we think this is a new word vector that contains all structural information

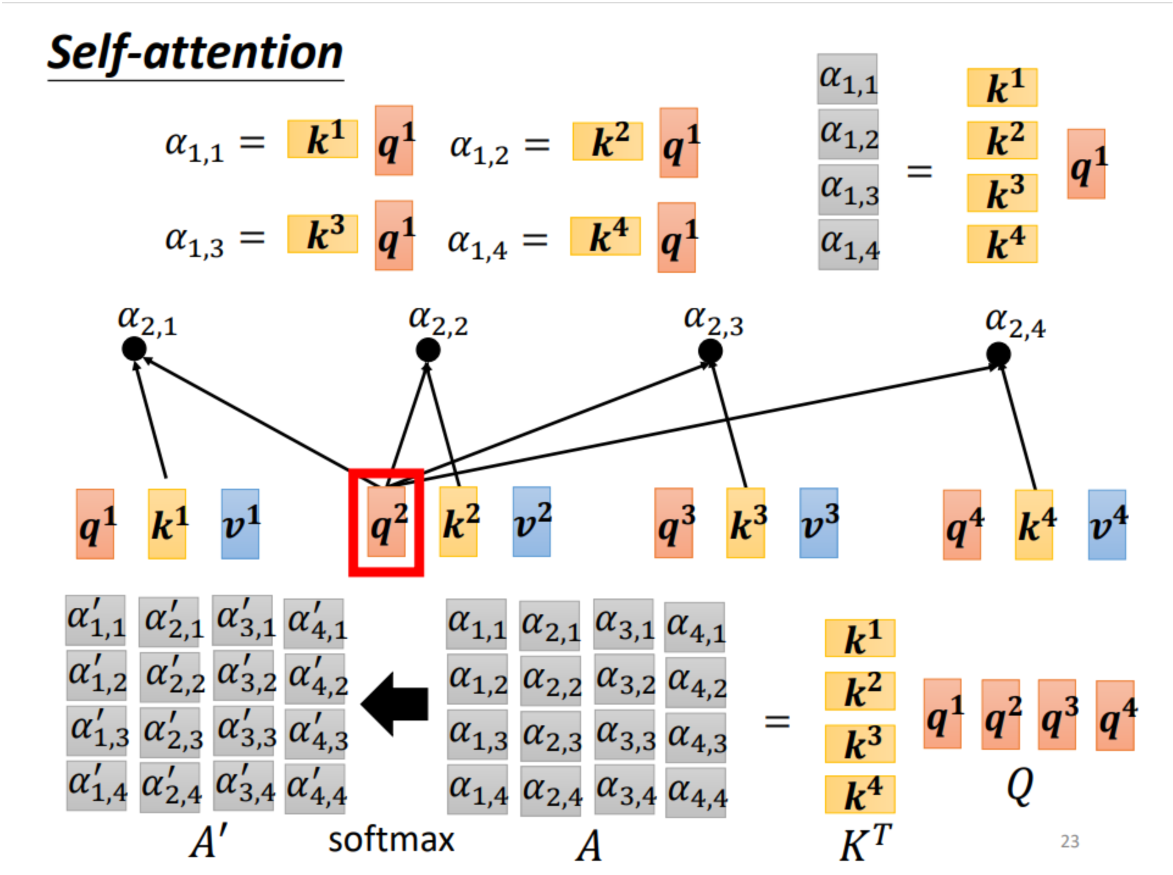
Cont.


Self-attention
$\alpha'_{1,i} = exp(\alpha_{1,i})/\sum_j exp(\alpha_{1,j})$
$\alpha'_{1,1}$ $\alpha'_{1,2}$ $\alpha'_{1,3}$ $\alpha'_{1,4}$
Soft-max
$\alpha_{1,1}$ $\alpha_{1,2}$ $\alpha_{1,3}$ $\alpha_{1,4}$
q^1 k^1 k^2 k^3 k^4
a^1 a^2 a^3 a^4
$q^1 = W^q a^1$
$k^1 = W^k a^1$
$k^2 = W^k a^2$
$k^3 = W^k a^3$
$k^4 = W^k a^4$
17

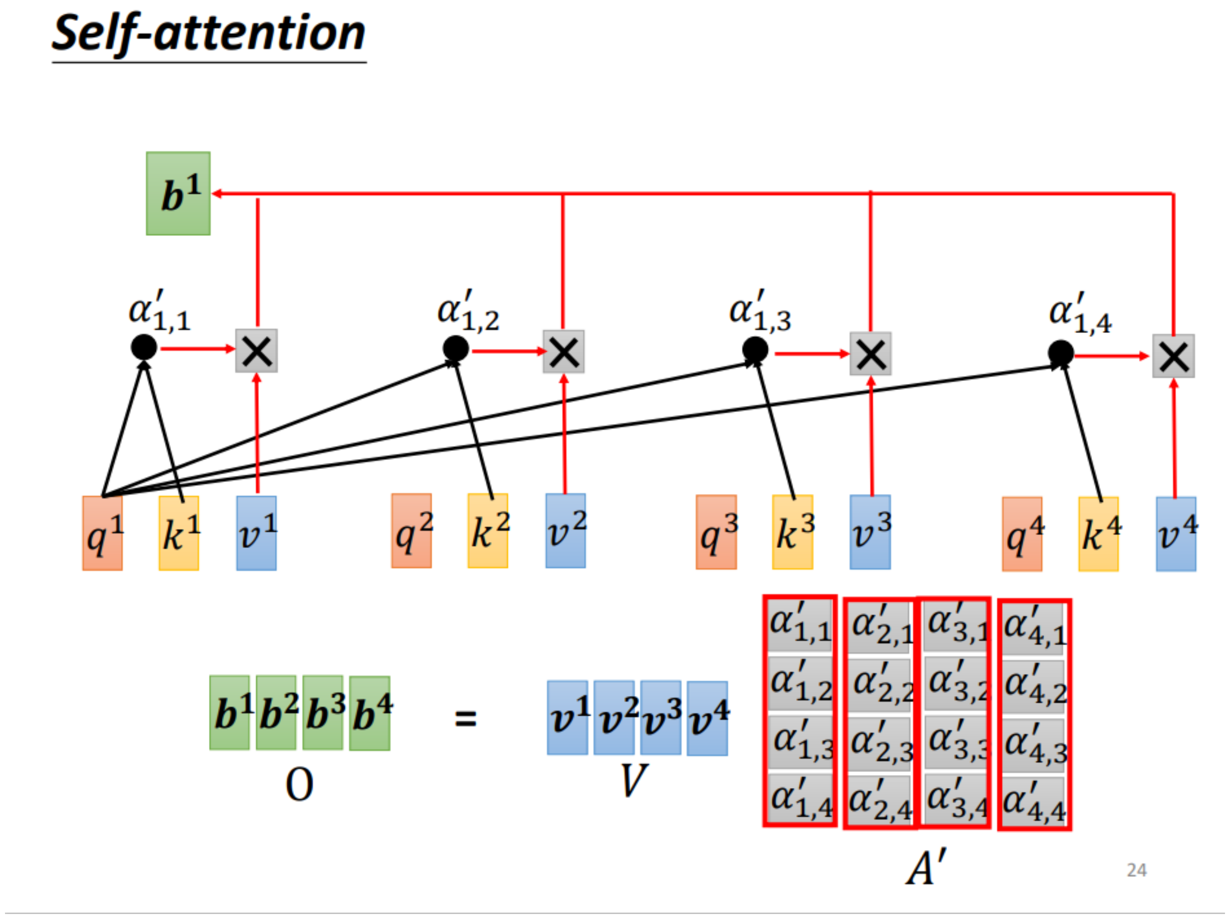

Self-attention
Extract information based on attention scores
$b^1 = \sum_i \alpha'_{1,i} v^i$
b^1
$\alpha'_{1,1}$ $\alpha'_{1,2}$ $\alpha'_{1,3}$ $\alpha'_{1,4}$
q^1 k^1 v^1 k^2 v^2 k^3 v^3 k^4 v^4
a^1 a^2 a^3 a^4
$v^1 = W^v a^1$
$v^2 = W^v a^2$
$v^3 = W^v a^3$
$v^4 = W^v a^4$
18

Cont.

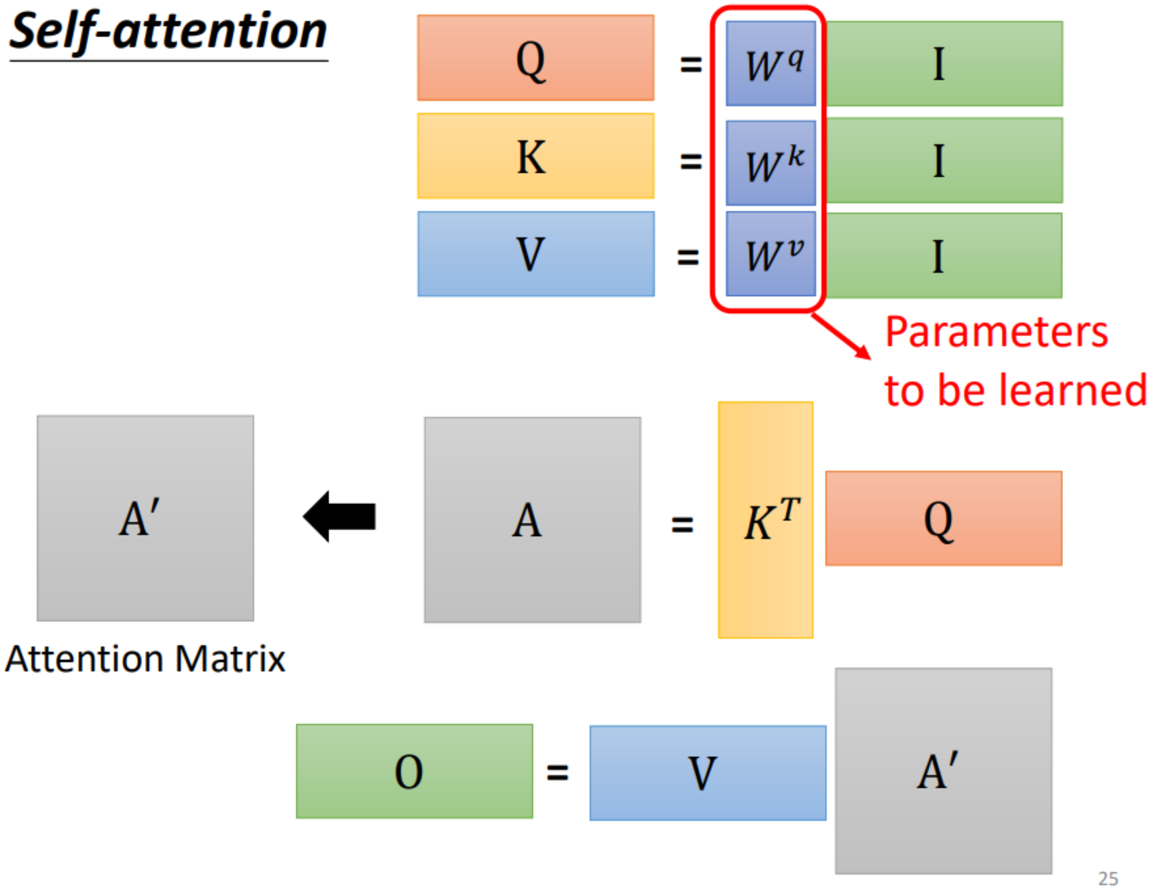

Self-attention
$q^i = W^q a^i$
$k^i = W^k a^i$
$v^i = W^v a^i$
$q^1 q^2 q^3 q^4$ = W^q $a^1 a^2 a^3 a^4$
Q
I
$k^1 k^2 k^3 k^4$ = W^k $a^1 a^2 a^3 a^4$
K
I
$v^1 v^2 v^3 v^4$ = W^v $a^1 a^2 a^3 a^4$
V
I
q^1 k^1 v^1
a^1
q^2 k^2 v^2
a^2
q^3 k^3 v^3
a^3
q^4 k^4 v^4
a^4
21


Self-attention
$\alpha_{1,1}$ = k^1 q^1
$\alpha_{1,2}$ = k^2 q^1
$\alpha_{1,3}$ = k^3 q^1
$\alpha_{1,4}$ = k^4 q^1
$\alpha_{1,1}$ $\alpha_{1,2}$ $\alpha_{1,3}$ $\alpha_{1,4}$ = k^1 k^2 k^3 k^4 q^1
$\alpha_{2,1}$ $\alpha_{2,2}$ $\alpha_{2,3}$ $\alpha_{2,4}$
q^1 k^1 v^1 q^2 k^2 v^2 q^3 k^3 v^3 q^4 k^4 v^4
$\alpha'_{1,1}$ $\alpha'_{2,1}$ $\alpha'_{3,1}$ $\alpha'_{4,1}$
$\alpha'_{1,2}$ $\alpha'_{2,2}$ $\alpha'_{3,2}$ $\alpha'_{4,2}$
$\alpha'_{1,3}$ $\alpha'_{2,3}$ $\alpha'_{3,3}$ $\alpha'_{4,3}$
$\alpha'_{1,4}$ $\alpha'_{2,4}$ $\alpha'_{3,4}$ $\alpha'_{4,4}$
A'
softmax
$\alpha_{1,1}$ $\alpha_{2,1}$ $\alpha_{3,1}$ $\alpha_{4,1}$
$\alpha_{1,2}$ $\alpha_{2,2}$ $\alpha_{3,2}$ $\alpha_{4,2}$
$\alpha_{1,3}$ $\alpha_{2,3}$ $\alpha_{3,3}$ $\alpha_{4,3}$
$\alpha_{1,4}$ $\alpha_{2,4}$ $\alpha_{3,4}$ $\alpha_{4,4}$
A
=
k^1 k^2 k^3 k^4
K^T
q^1 q^2 q^3 q^4
Q
23

Cont.



Self-attention
b^1
$\alpha'_{1,1}$
$\alpha'_{1,2}$
$\alpha'_{1,3}$
$\alpha'_{1,4}$
q^1 k^1 v^1 q^2 k^2 v^2 q^3 k^3 v^3 q^4 k^4 v^4
$b^1\ b^2\ b^3\ b^4$ = $v^1\ v^2\ v^3\ v^4$
O
V
A'
24



Self-attention
Q = W^q I
K = W^k I
V = W^v I
Parameters to be learned
A′
A = K^T Q
Attention Matrix
O = V A′
25

Position encoding

- Self-attention only considers the content of the context and does not consider the relationship between text positions
- “I love you” and “you love I” have the same output using Self-attention

- Solution:
- 1. add a position embedding to the original input x
- 2. Using a trainable position embedding parameter

Cont.

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

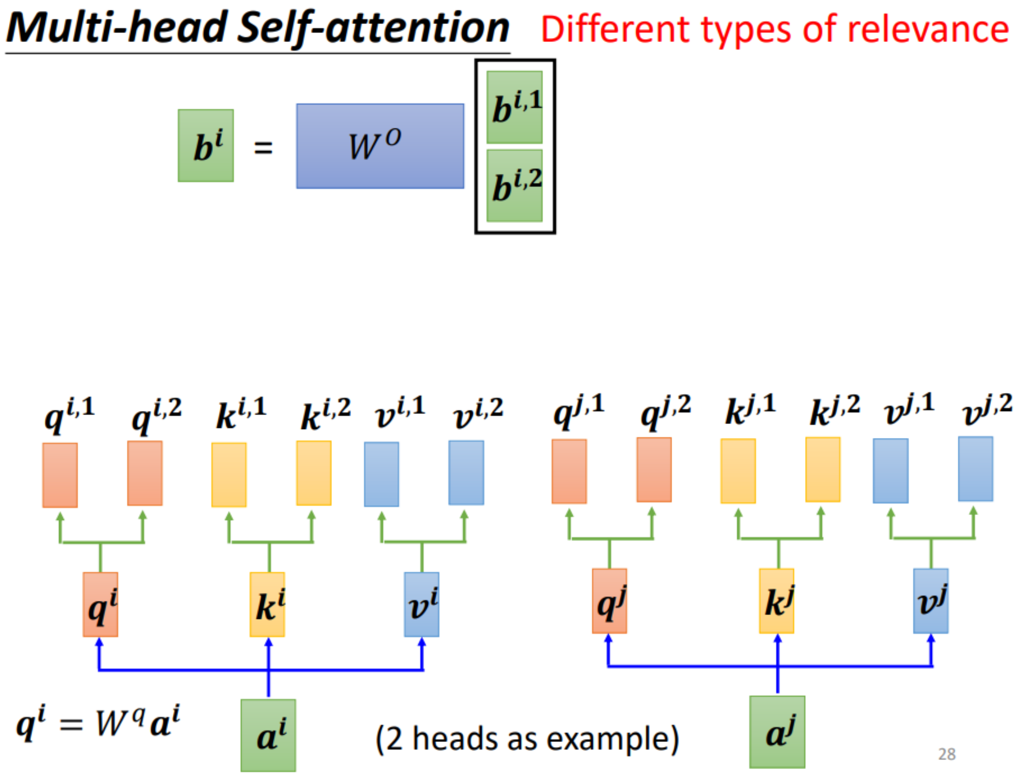
- Add position embedding to the sentence “I love you”
- Suppose: variable “pos” starts from 0, dmodel = 4 (that is, dimension of the word vector)

word	Dimension 1	Dimension 2	Dimension 3	Dimension 4
I	0	0	0	0
love	$\sin(1/10000^{2/4})$	$\cos(1/10000^{2/4})$		
you				

- For word “I” :
  - pos = 0
  - i = 0: PE(0, 0) = sin(0/whatever) = 0; PE(0, 0) = 0
  - i =1: PE(0, 2) = 0; PE(0, 3) = 0

Multi-head Attention

$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(\mathrm{head_1}, \ldots, \mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$



- Suppose:
- Size X = (4, 2) $[x_1, x_2]^T$
- Size QKV = (2, 2) W = (2, 4) Dimension reduction
- If num of head => 2
- For q_1 split into 2 sub-matrix q_{11} and q_{12}
- q_{11}: (2, 1) q_{12}: (2, 1) k_{11}: (2, 1) k_{12}: (2, 1)
- $a_{11} = k^T_{11} * q_{11}$ = (1, 2)*(2, 1) = (1, 1) then Softmax
- $b_{11} = v_{11} \cdot a'_{11}$ = (2, 1) · (1, 1) = (2, 1)
- Combine b_{11} b_{12} we get => (4, 1)
  - Also for b_{21} and b_{22} we get (4, 1) add, then (4, 2)
- Optimize through a fully connected (FC) layer

Interview questions

- What is the core of Self-Attention?
- Why does Transformer require additional positional encoding?
- What exactly is QKV matrix in Self-attention? Why do they exist?
- Why does Transformer need Multi-head Attention? What are its benefits?
- Why must the dot product model be scaled before Softmax normalization?
- Why does each head in multi-head attention in Transformer need to be dimensionally reduced?