Transformer model

Presenter: Steven Tang

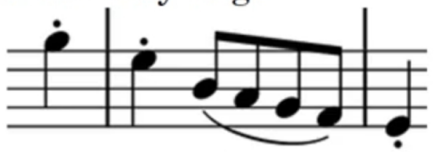
2021.09

What I will cover

- The motivation behind transformer model
- The architecture of transformer model
- Multi-head self-attention mechanism (What is Q,K,V?)
- Implementation considerations
- Very brief about training

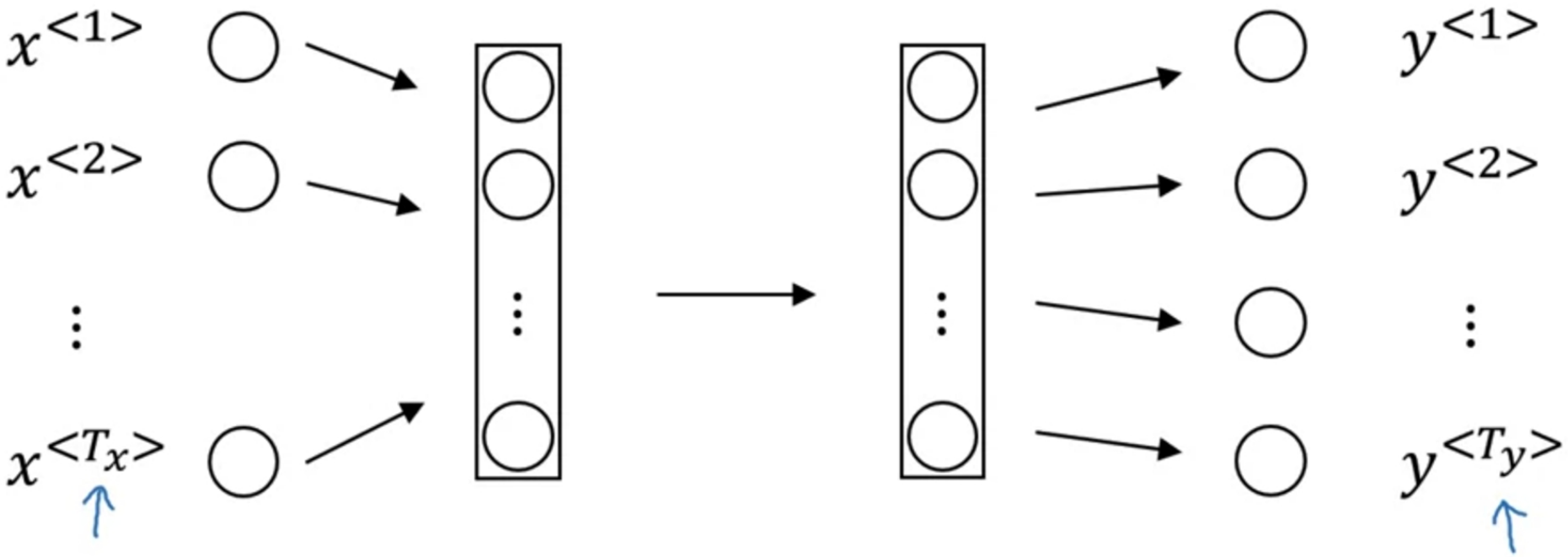
Sequence models use-case

- ML Models that works on sequence data

Speech recognition		"The quick brown fox jumped over the lazy dog."
Music generation		
Sentiment classification	"There is nothing to like in this movie."	★☆☆☆☆
DNA sequence analysis	AGCCCCTGTGAGGAACTAG	AGCCCCTGTGAGGAACTAG
Machine translation	Voulez-vous chanter avec moi?	Do you want to sing with me?
Video activity recognition		Running
Name entity recognition	Yesterday, Harry Potter met Hermione Granger.	Yesterday, Harry Potter met Hermione Granger.

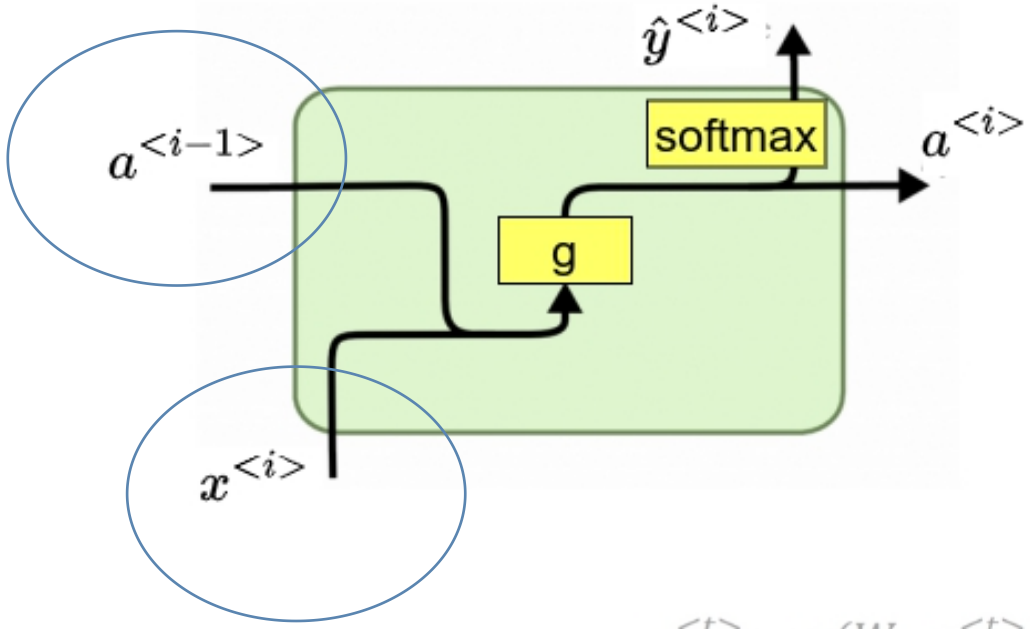
Wny not a standard network?

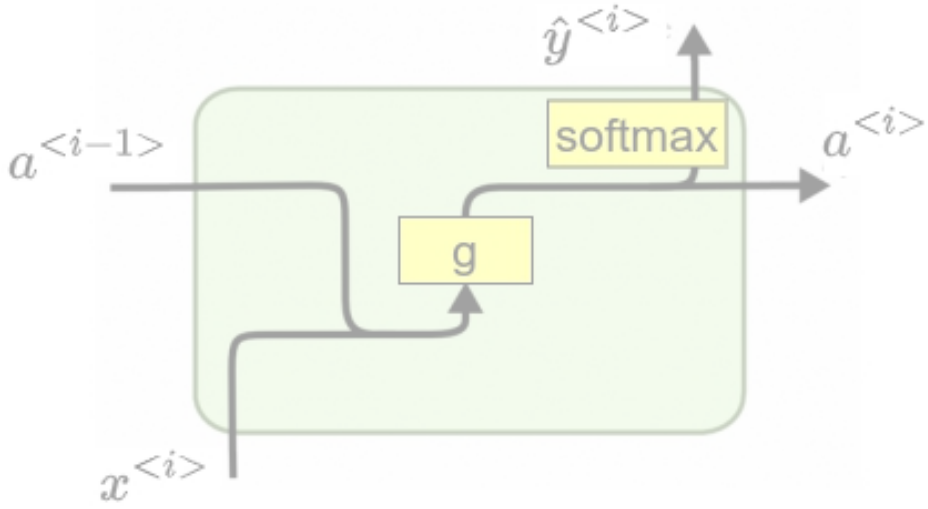
- 1. Variable length inputs
- 2. No learned feature sharing



RNN cells

RNN GRU LSTM



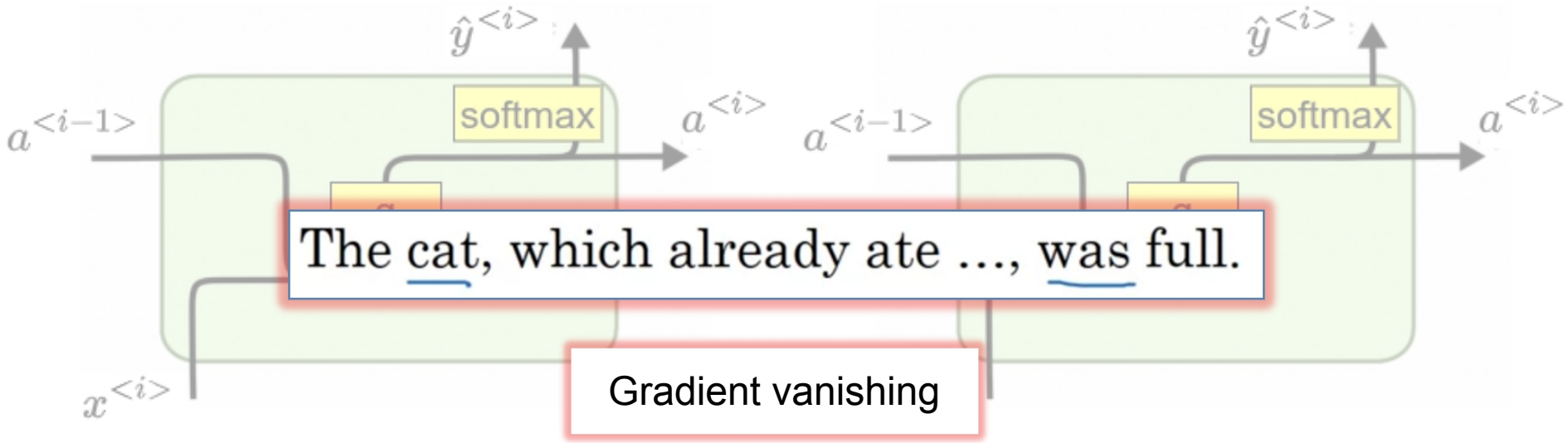


$$a^{<t>} = g(W_{aa}a^{<t>} + W_{ax}x^{<t>} + b_a)$$

$$\hat{y}^{<t>} = g(W_{ya}a^{<t>} + b_y)$$

RNN cells

RNN GRU LSTM

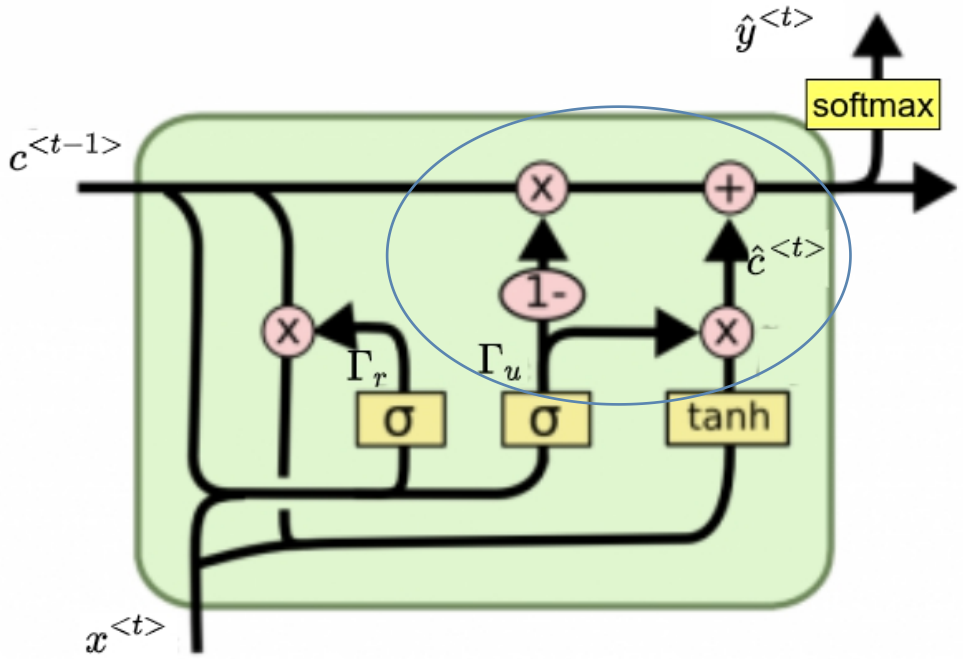


$$a^{<t>} = g(W_{aa}a^{<t>} + W_{ax}x^{<t>} + b_a)$$

$$\hat{y}^{<t>} = g(W_{ya}a^{<t>} + b_y)$$

RNN cells

RNN GRU LSTM



$$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c)$$

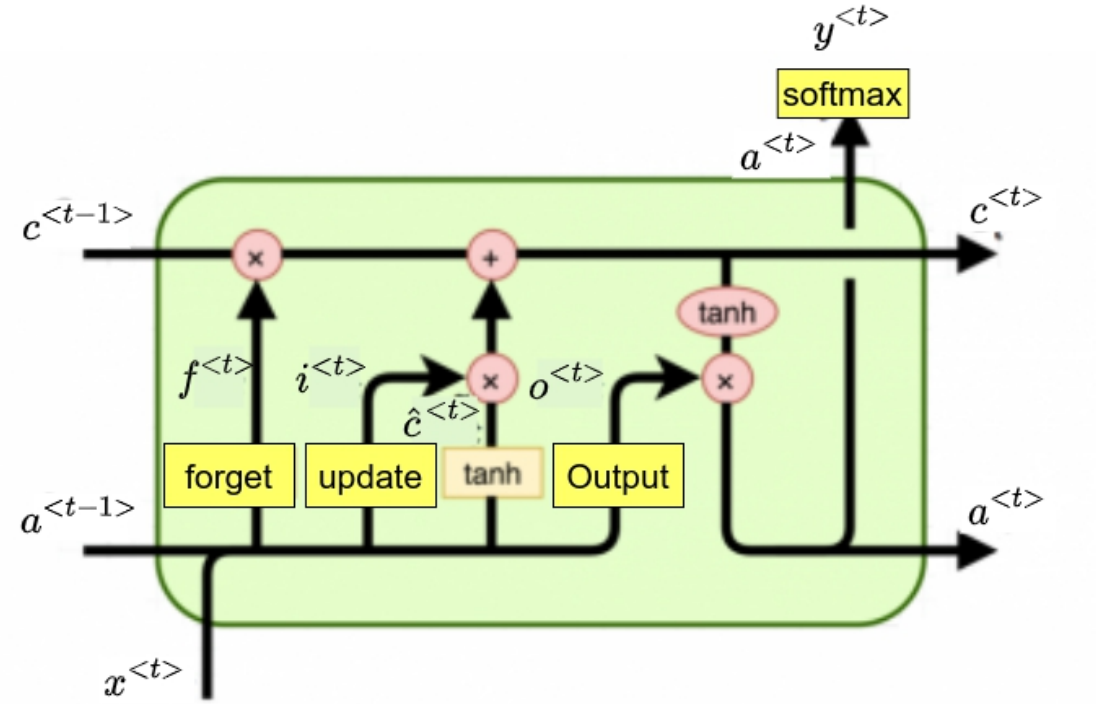
$$\Gamma_u = \sigma(W_u[c^{<t-1>}, x^{<t>}] + b_u)$$

$$\Gamma_r = \sigma(W_r[c^{<t-1>}, x^{<t>}] + b_r)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) + c^{<t-1>}$$

RNN cells

RNN GRU LSTM



$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u)$$

$$\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f)$$

$$\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o)$$

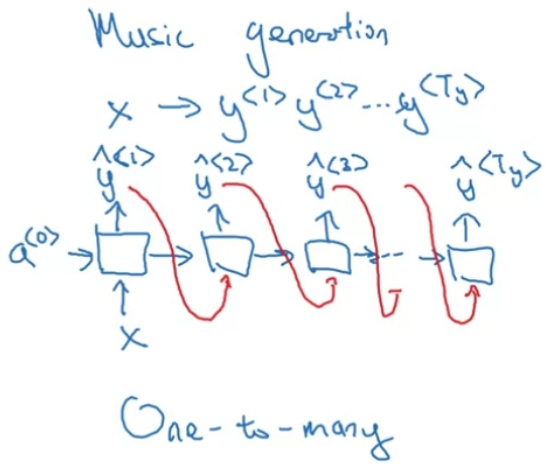
$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$

$$a^{<t>} = \Gamma_o * \tanh c^{<t>}$$

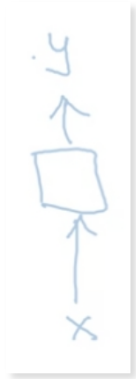
The cat, which already ate …, was full.

Different types of RNNs

- One-to-many

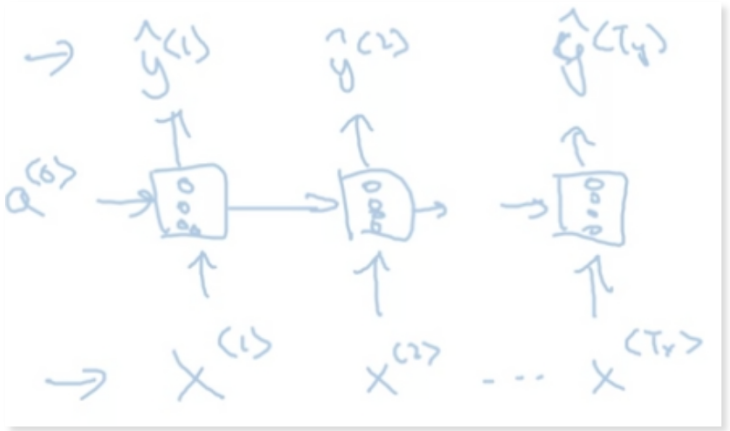


- One-to-one

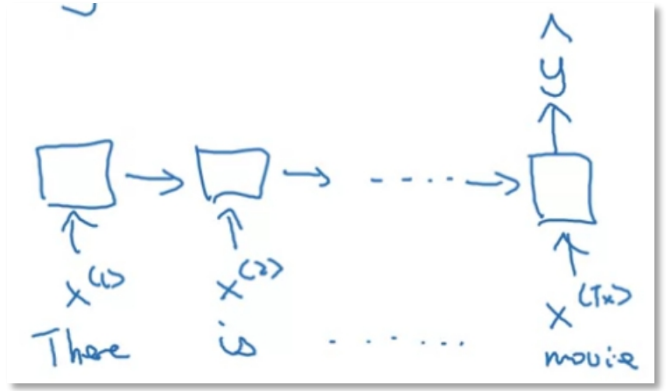


.....

- Many-to-many

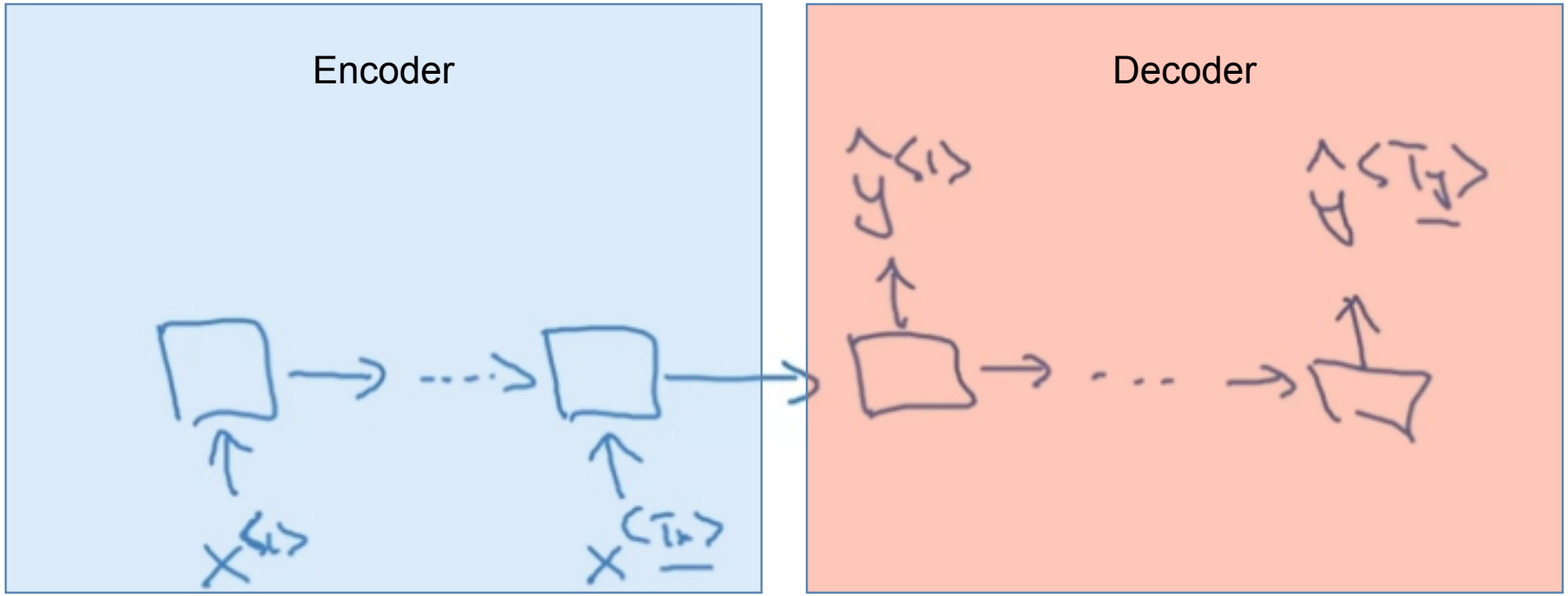


- Many-to-one



Encoder-Decoder architecture

- Many-to-many



Encoder-Decoder architecture

$x^{<1>}$ $x^{<2>}$ $x^{<3>}$ $x^{<4>}$ $x^{<5>}$

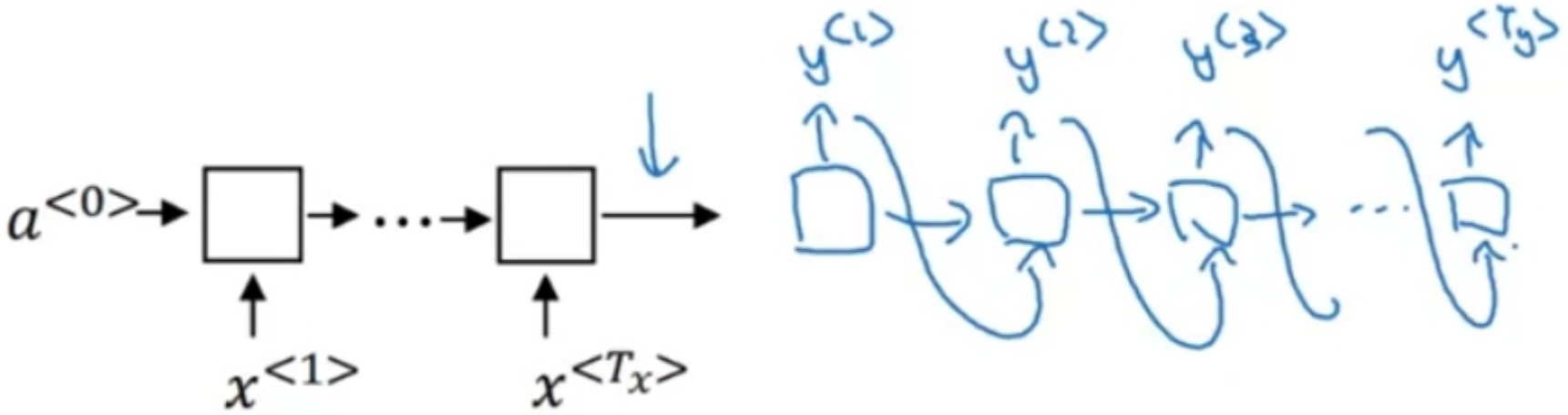
Jane visite L'Afrique en Septembre

-->

Jane is visiting Africa in September

$y^{<1>}$ $y^{<2>}$ $y^{<3>}$ $y^{<4>}$ $y^{<5>}$ $y^{<6>}$

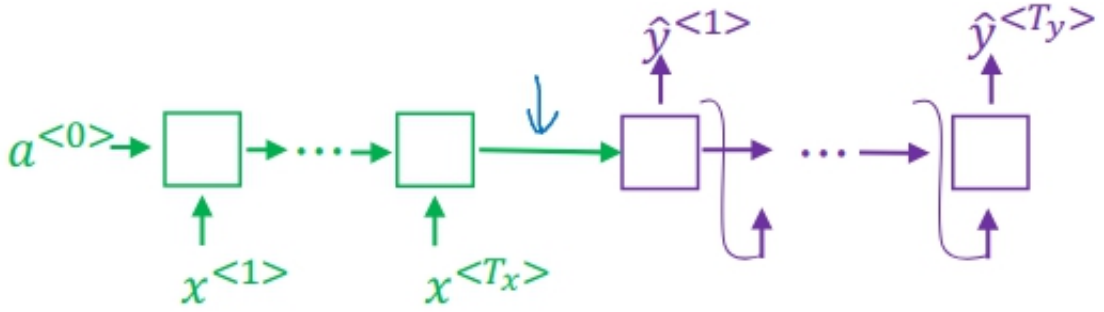
Seq2Seq Problem

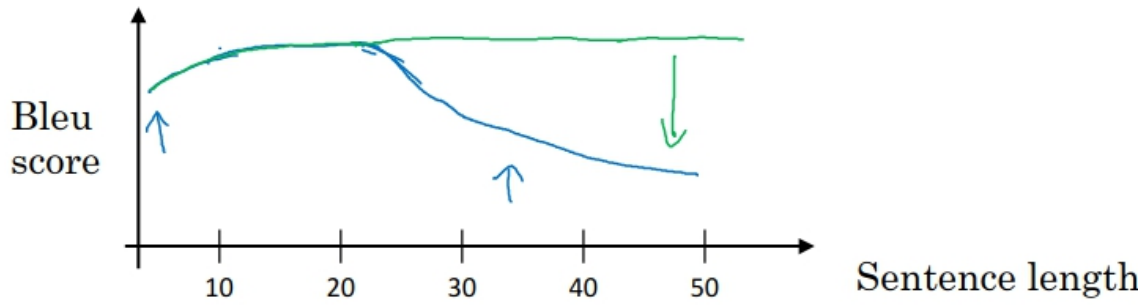


Common solution

It is not perfect

- Jane s'est rendue en Afrique en septembre dernier, a apprécié la culture et a rencontré beaucoup degens merveilleux; elle est revenue en parlant comment son voyage était merveilleux, et elle me tented'y aller aussi.
- Jane went to Africa last September, and enjoyed the culture and met many wonderful people; she came back raving about how wonderful her trip was, and is tempting me to go too.



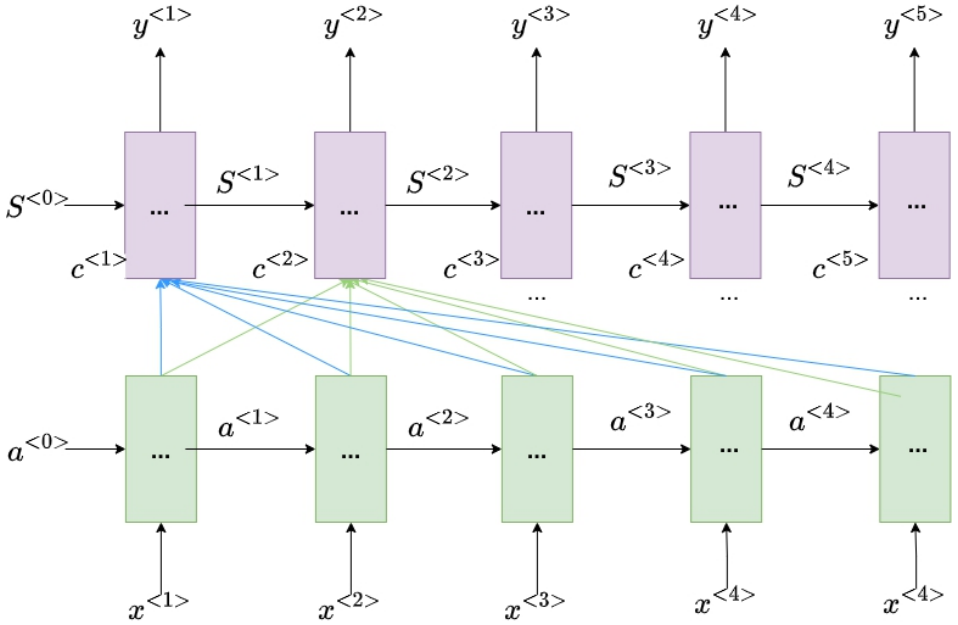


How does human solve the problem?

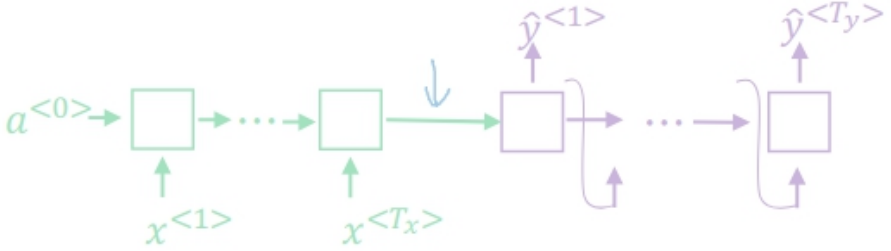
- Jane s'est rendue en Afrique en septembre dernier, a apprécié la culture et a rencontré beaucoup degens merveilleux; elle est revenue en parlant comment son voyage était merveilleux, et elle me tented'y aller aussi.
- Jane went to Africa last September, and enjoyed the culture and met many wonderful people; she came back raving about how wonderful her trip was, and is tempting me to go too.

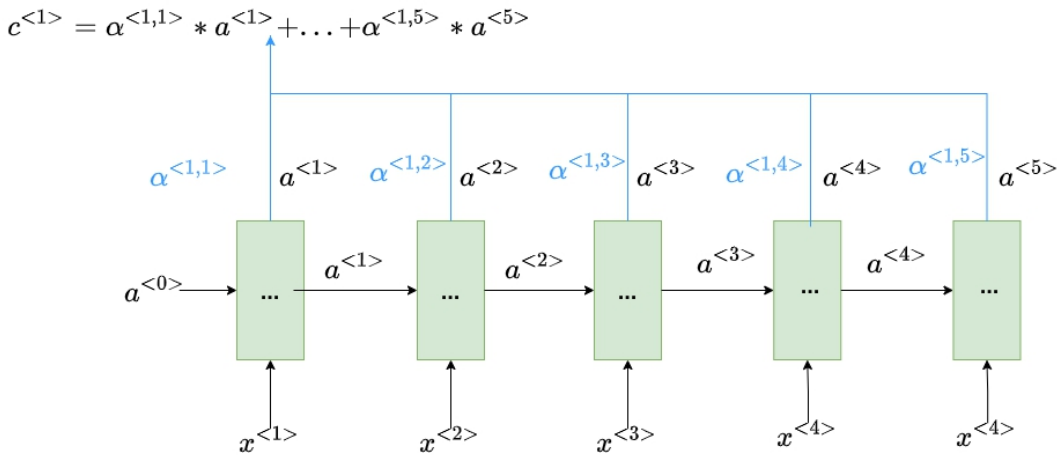
“Attention” saves brain power

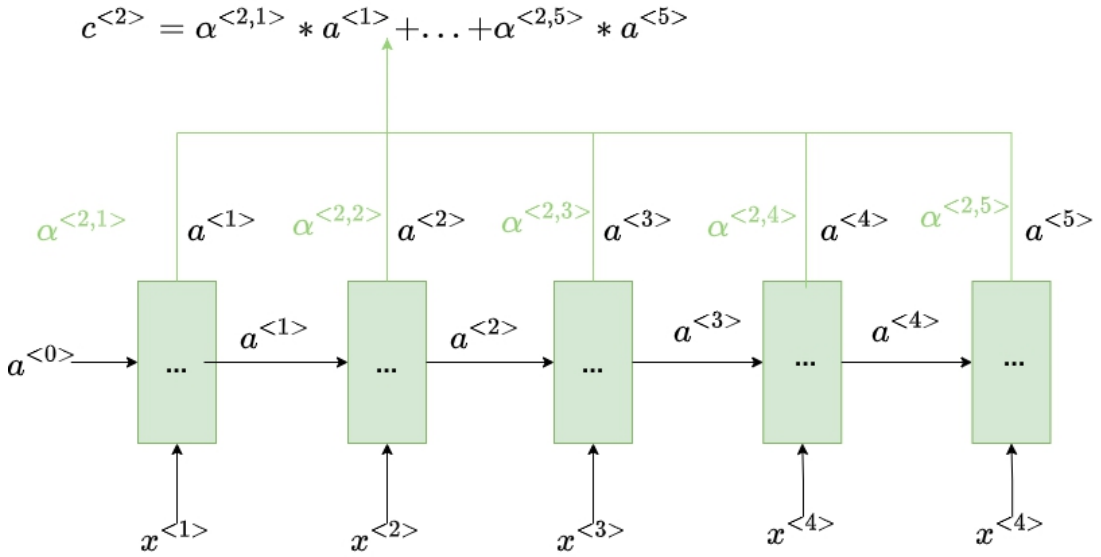
How to represent attention in Encoder-Decoder



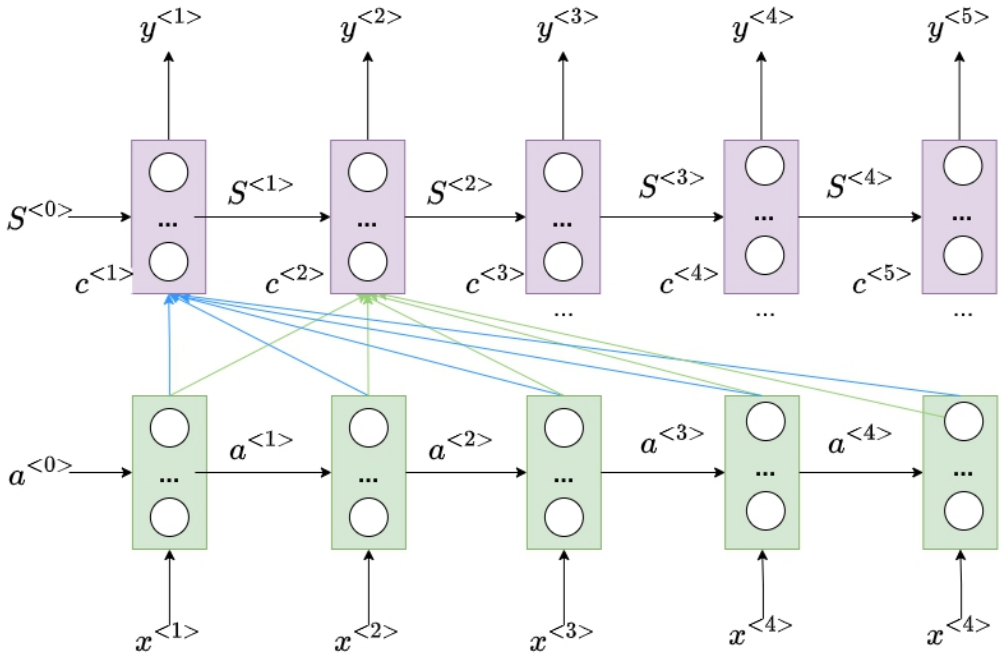
Jane visite L'Afrique en Septembre





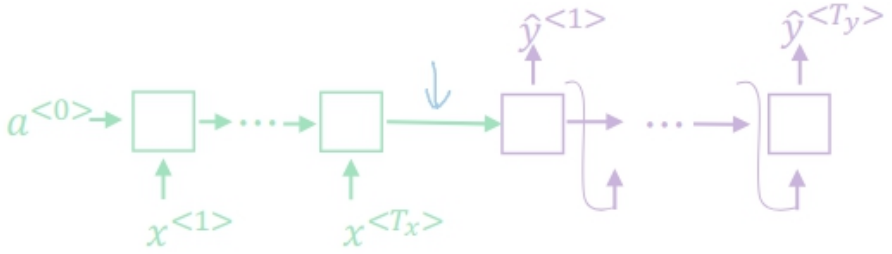


How to represent attention in Encoder-Decoder

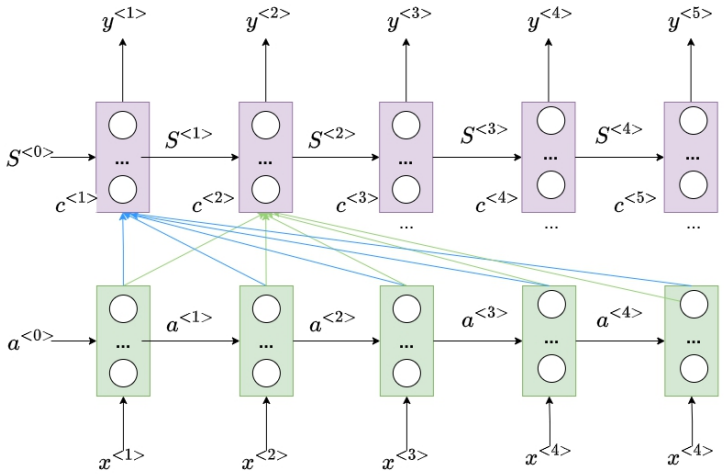


Jane visite L'Afrique en Septembre

$$c^{<t>} = \sum_{t'} \alpha^{<1,t'>} a^{<t'>}$$



How to calculate attention?



$$c^{<t>} = \sum_{t'} \alpha^{<1,t'>} a^{<}$$

①

$$\alpha^{<t,t'>} = \frac{\exp(e^{<t,t'>})}{\sum_{t'=1}^{Tx} \exp(e^{<t,t'>})}$$

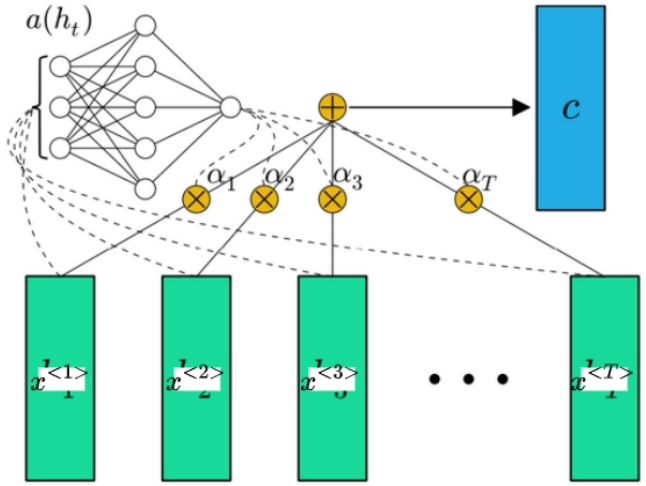
$s^{<t-1>}$

$a^{<t'>}$

$e^{<t,t'>}$

$\alpha^{<t,t'>}$

②



③

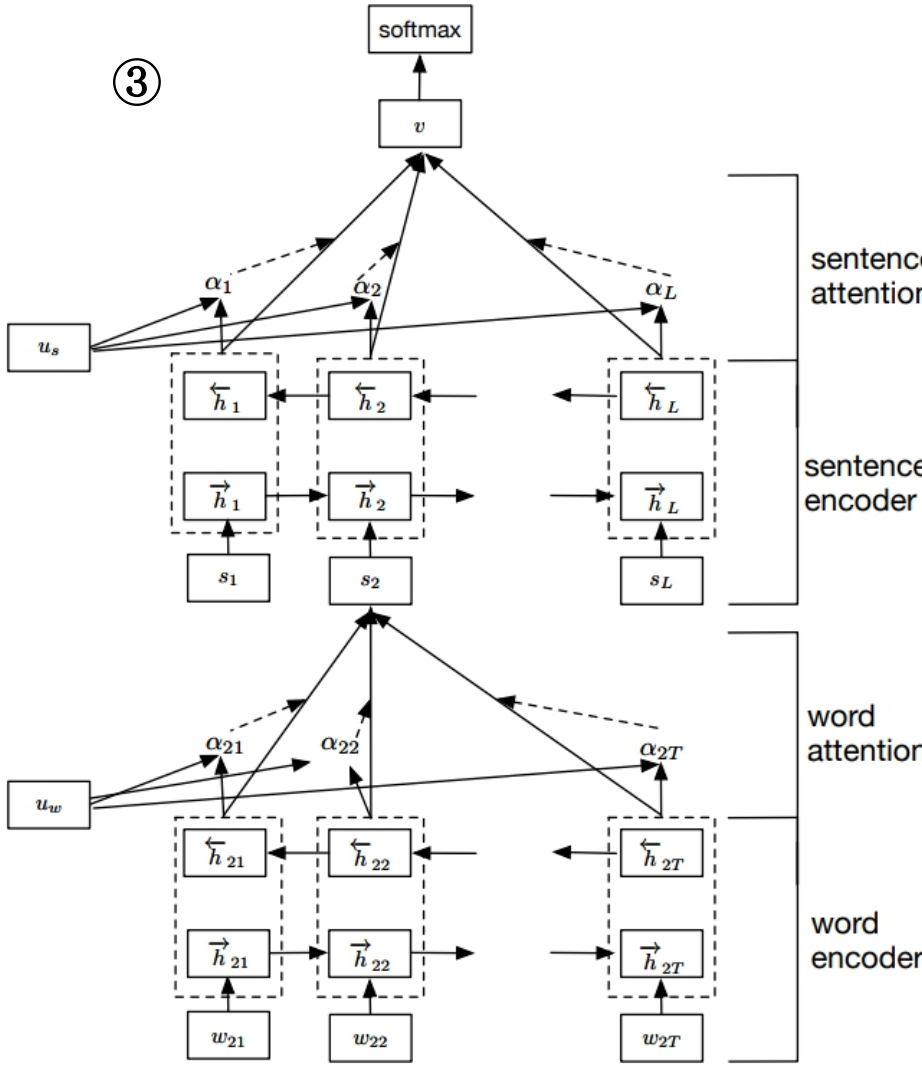
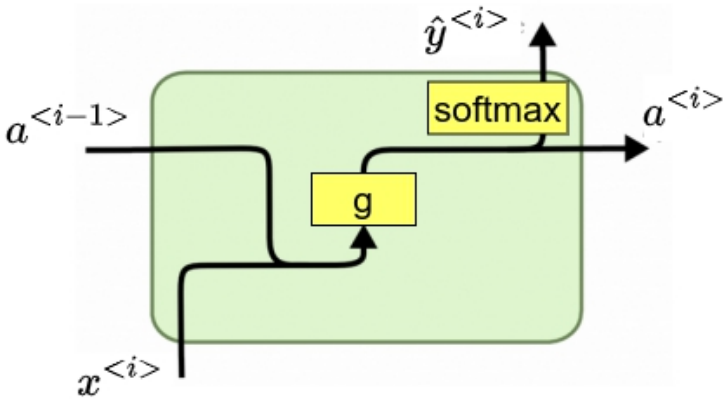


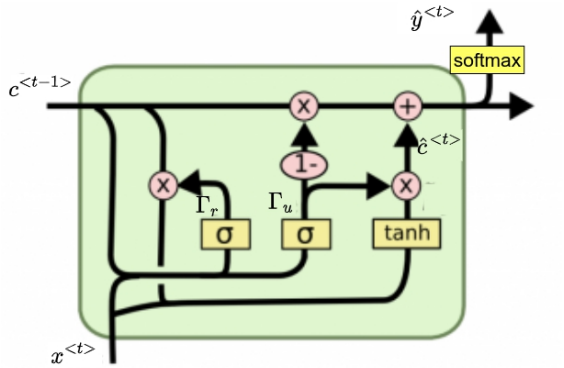
Figure 2: Hierarchical Attention Network.

Limitation?

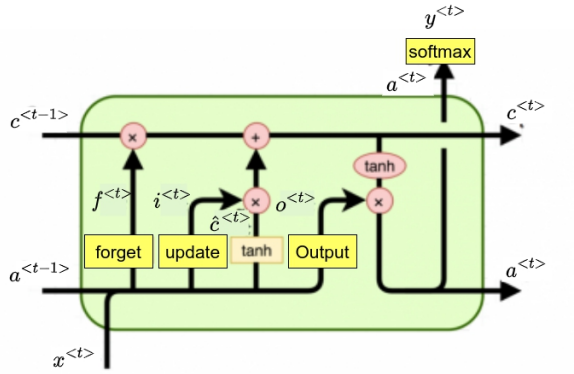
RNN



GRU



LSTM

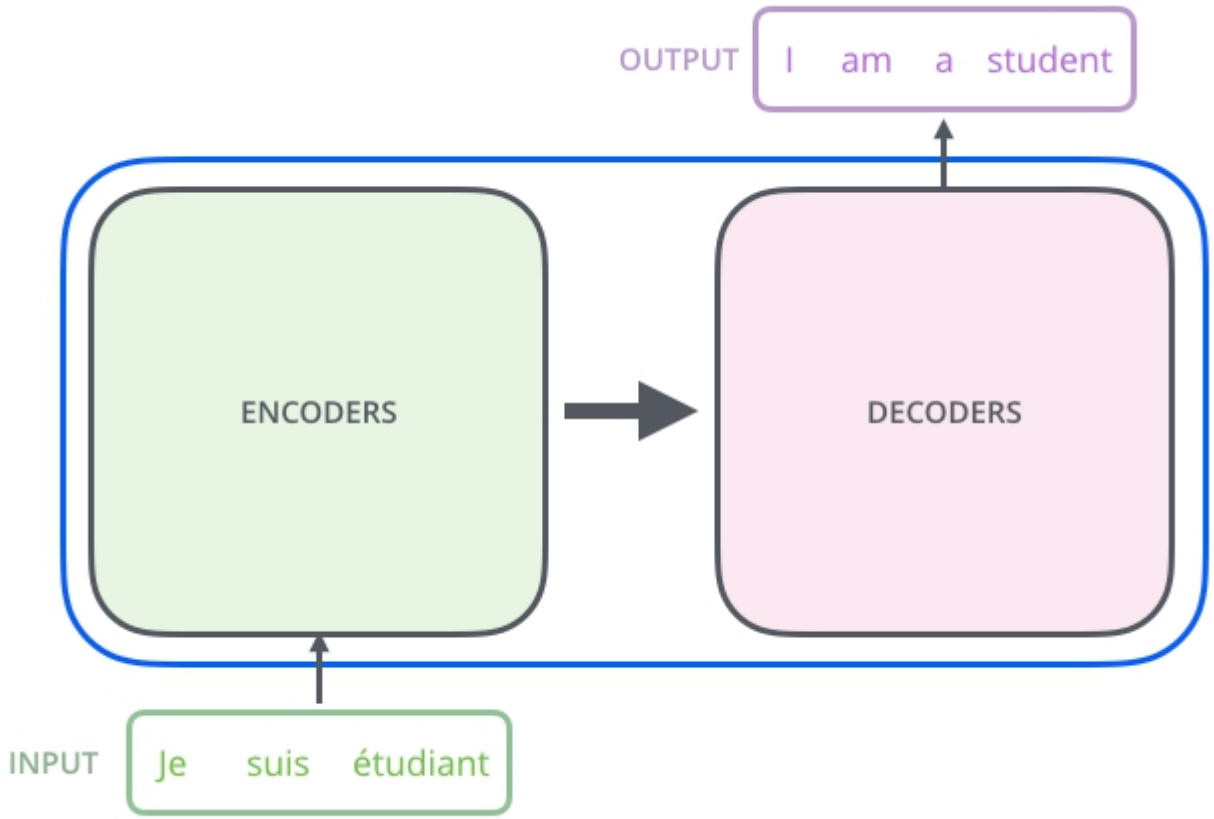


More and more complex

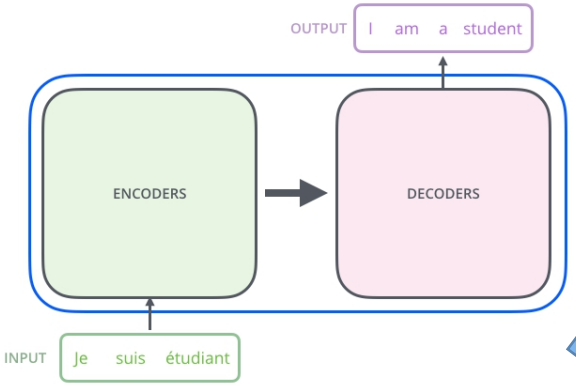
+

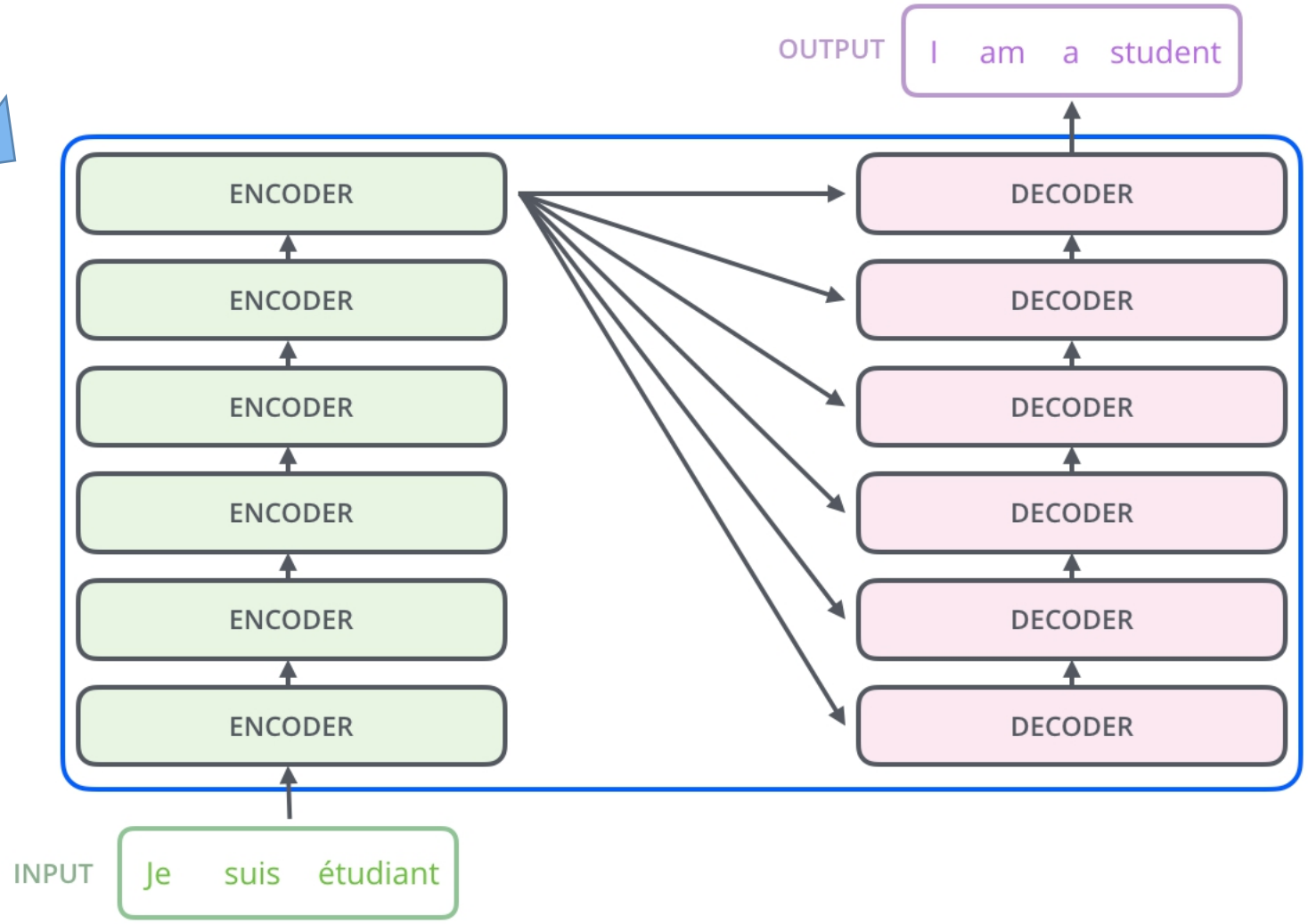
Sequential

Overall architecture



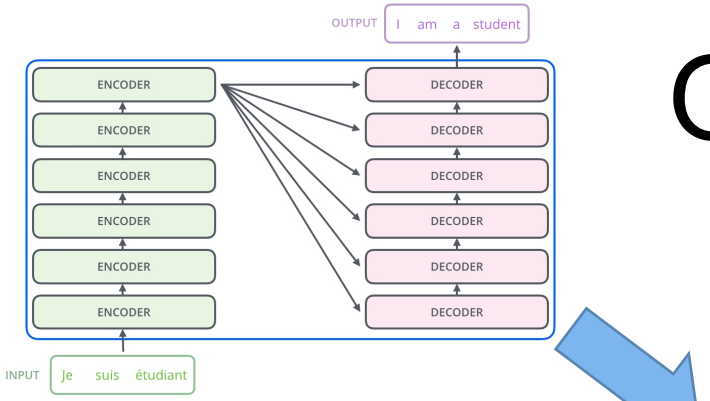
Overall architecture





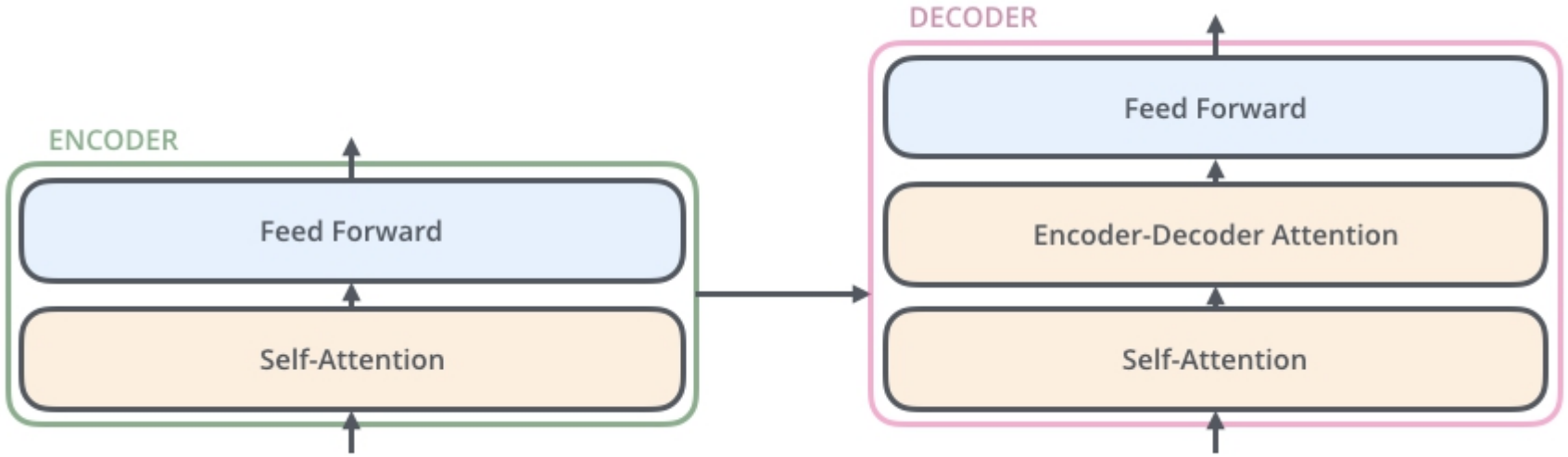
Encoder:
Same structure,
no weight sharing

Overall architecture



Self-Attention: Focus on other words in the input sentence

Encoder-Decoder Attention: Focus on relevant parts of the input sentence



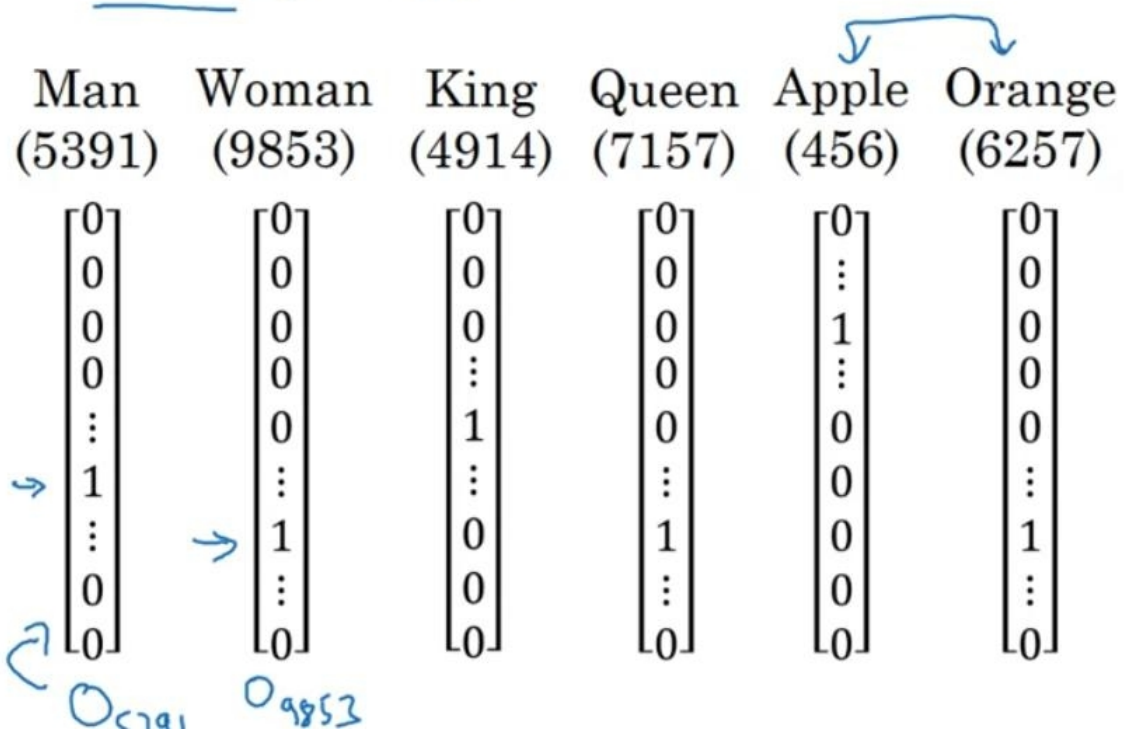
Bringing tensors into the picture

- Word embedding

V = [a, aaron, …, zulu, <UNK>]

|V| = 10,000

1-hot representation



I want a glass of orange juice.

I want a glass of apple ?.

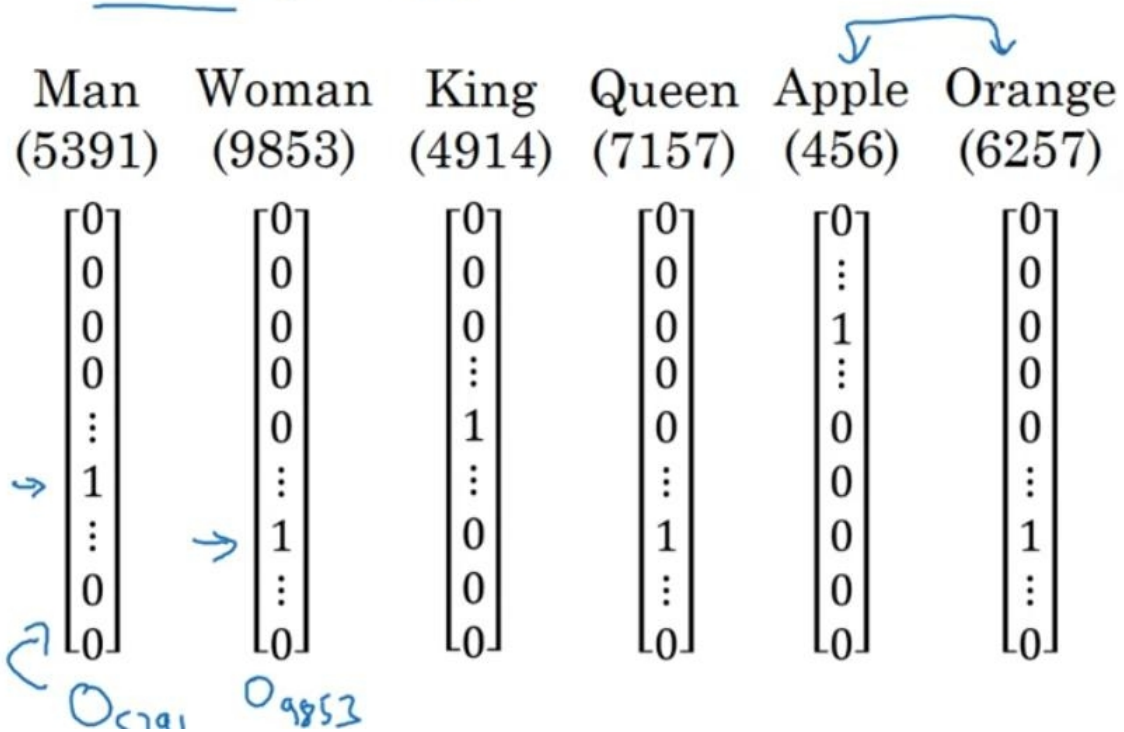
Bringing tensors into the picture

- Word embedding

V = [a, aaron, …, zulu, <UNK>]

|V| = 10,000

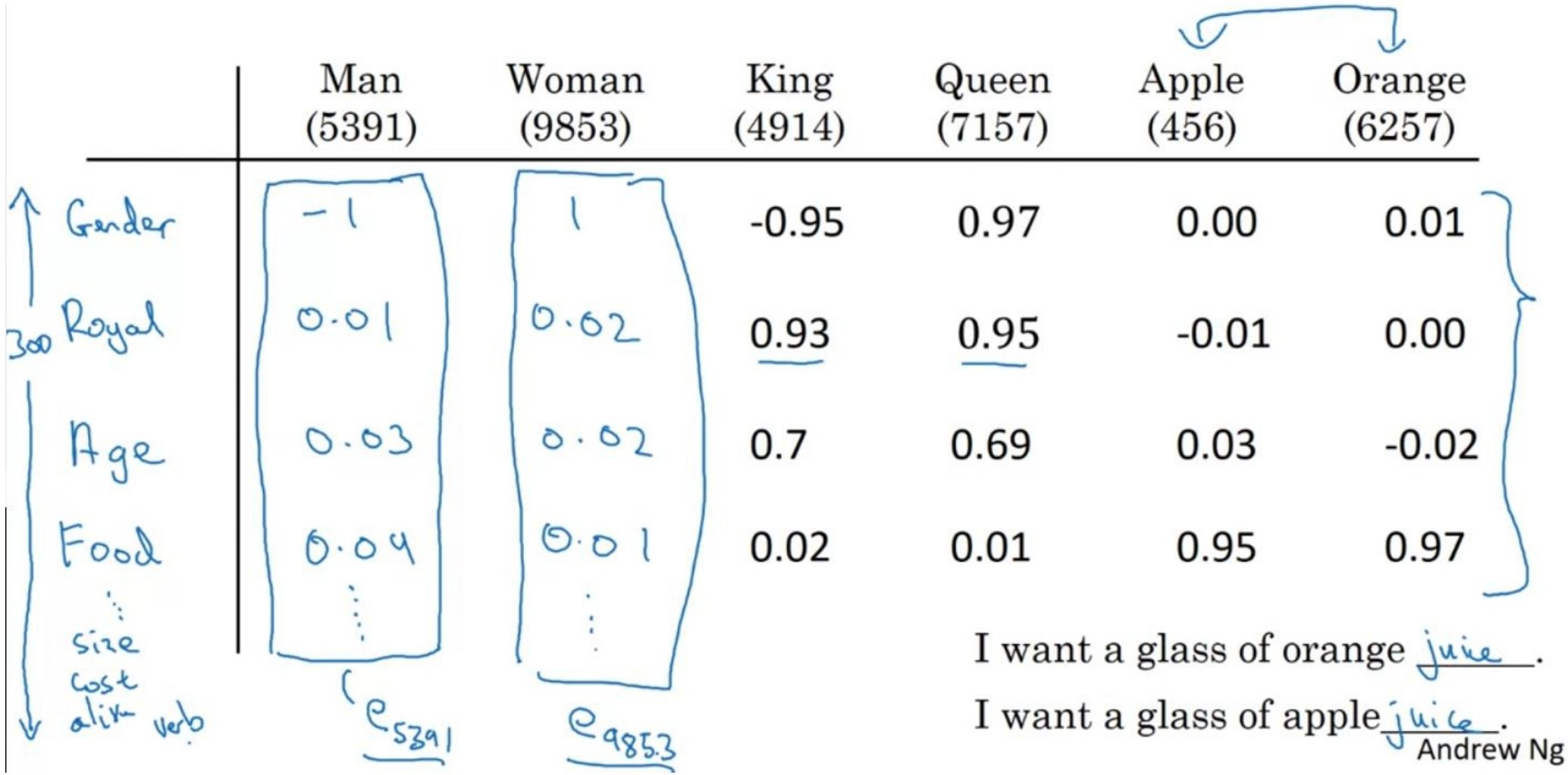
1-hot representation



I want a glass of orange juice.

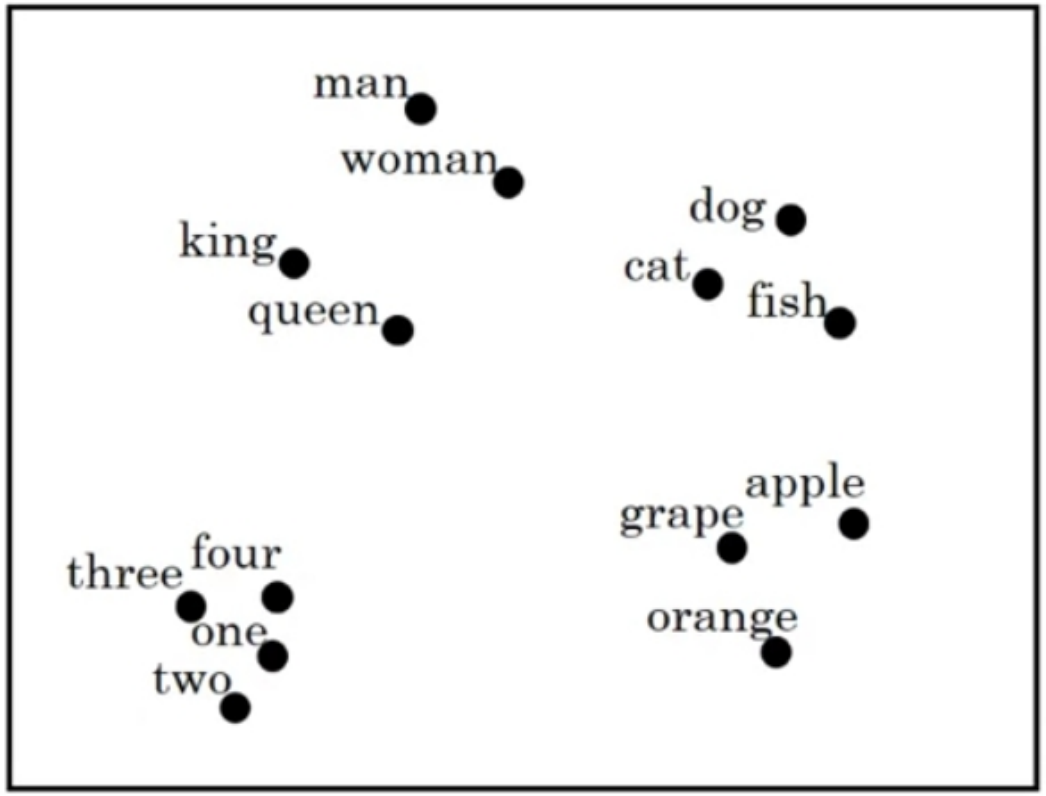
I want a glass of apple ?.

Bringing tensors into the picture

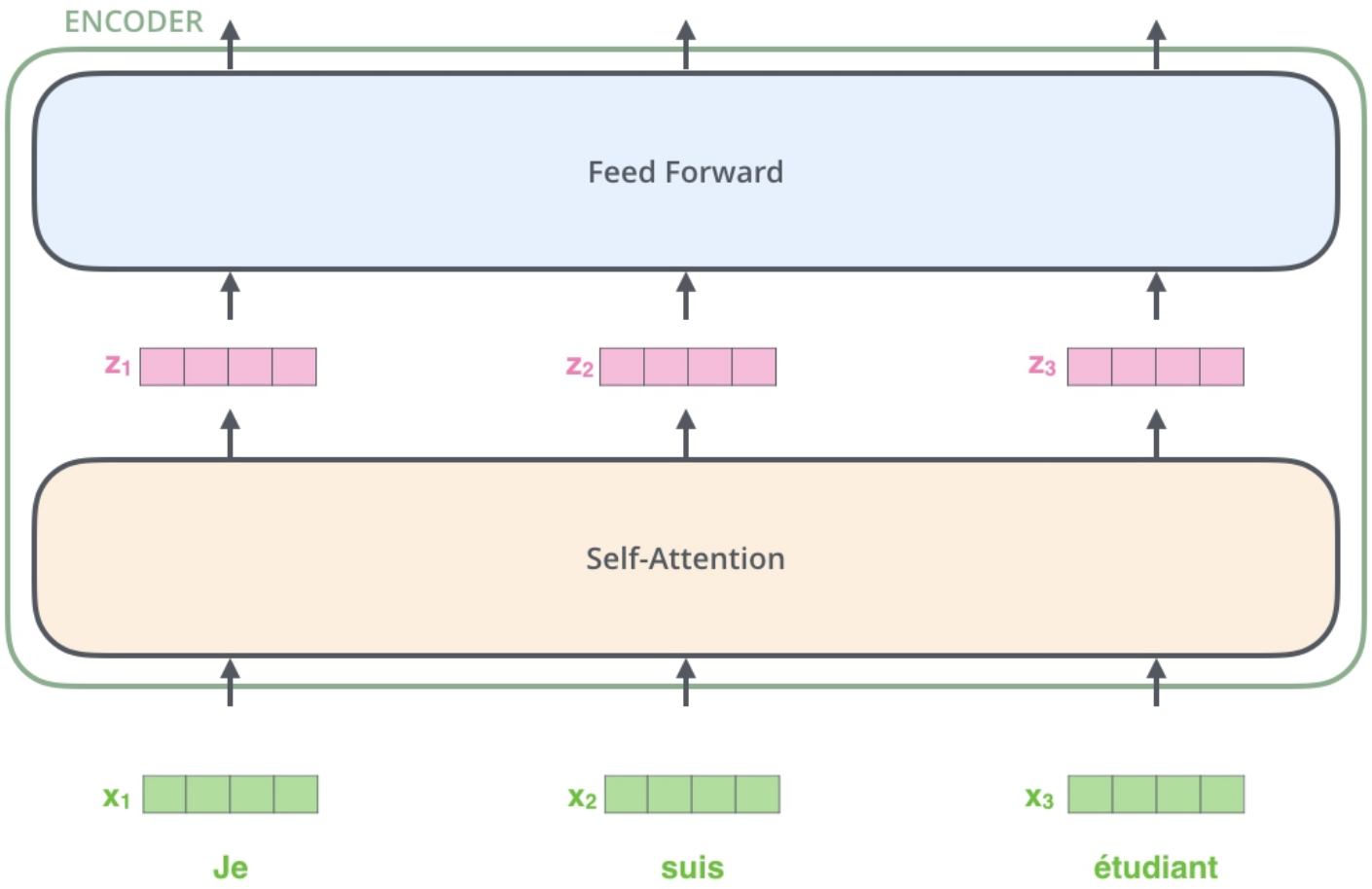


Andrew Ng

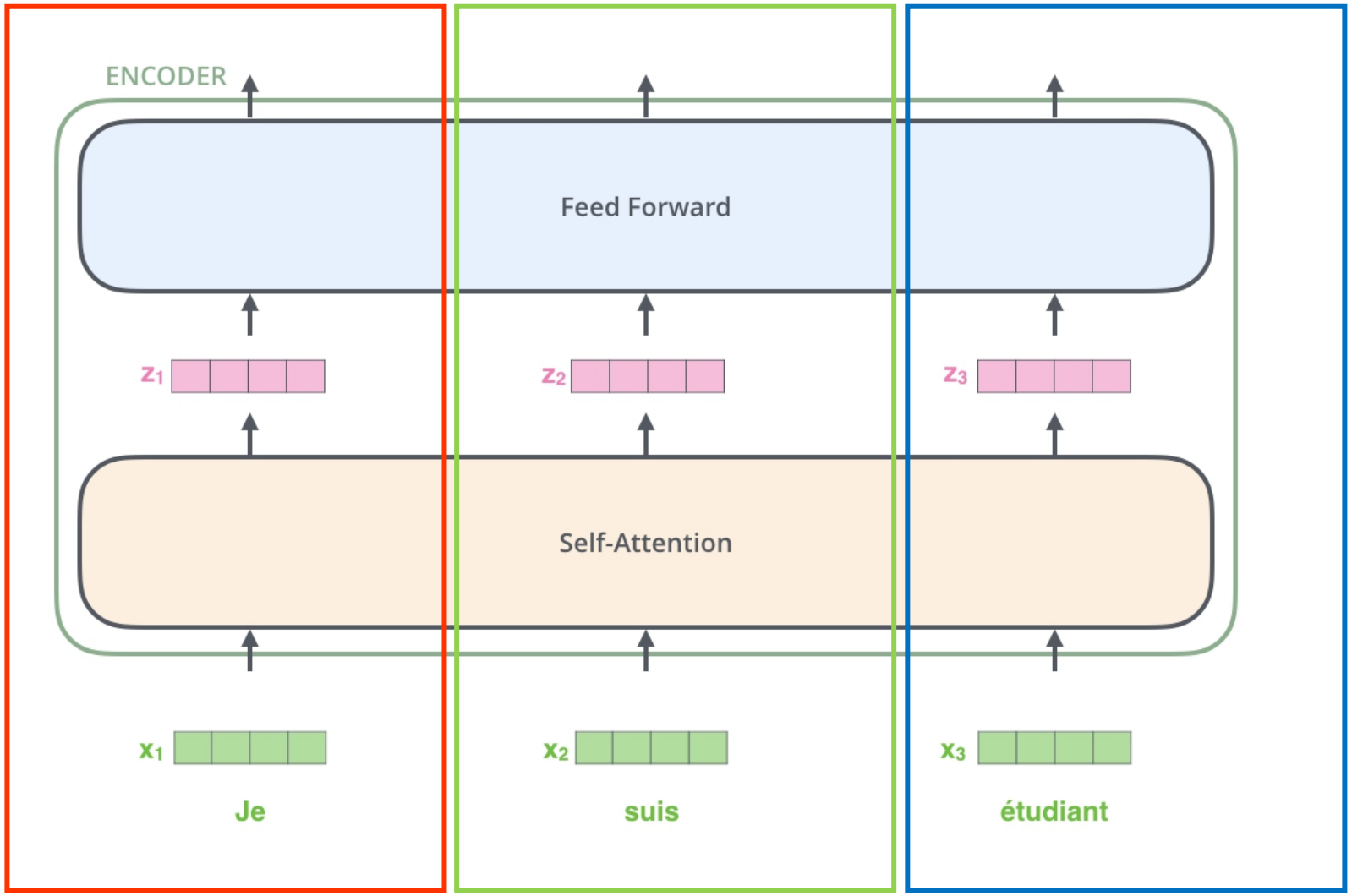
Bringing tensors into the picture



Bringing tensors into the picture



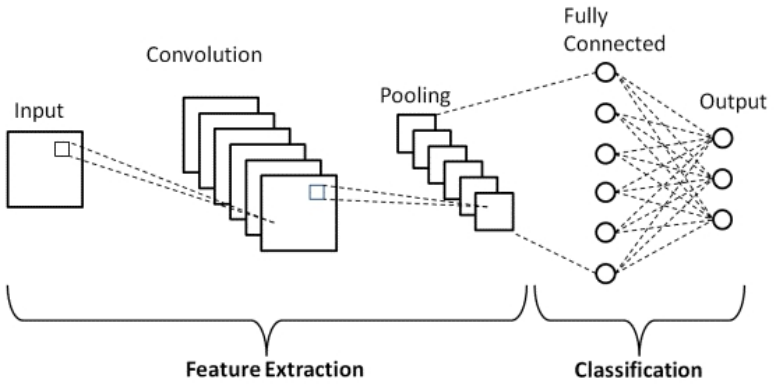
Bringing tensors into the picture

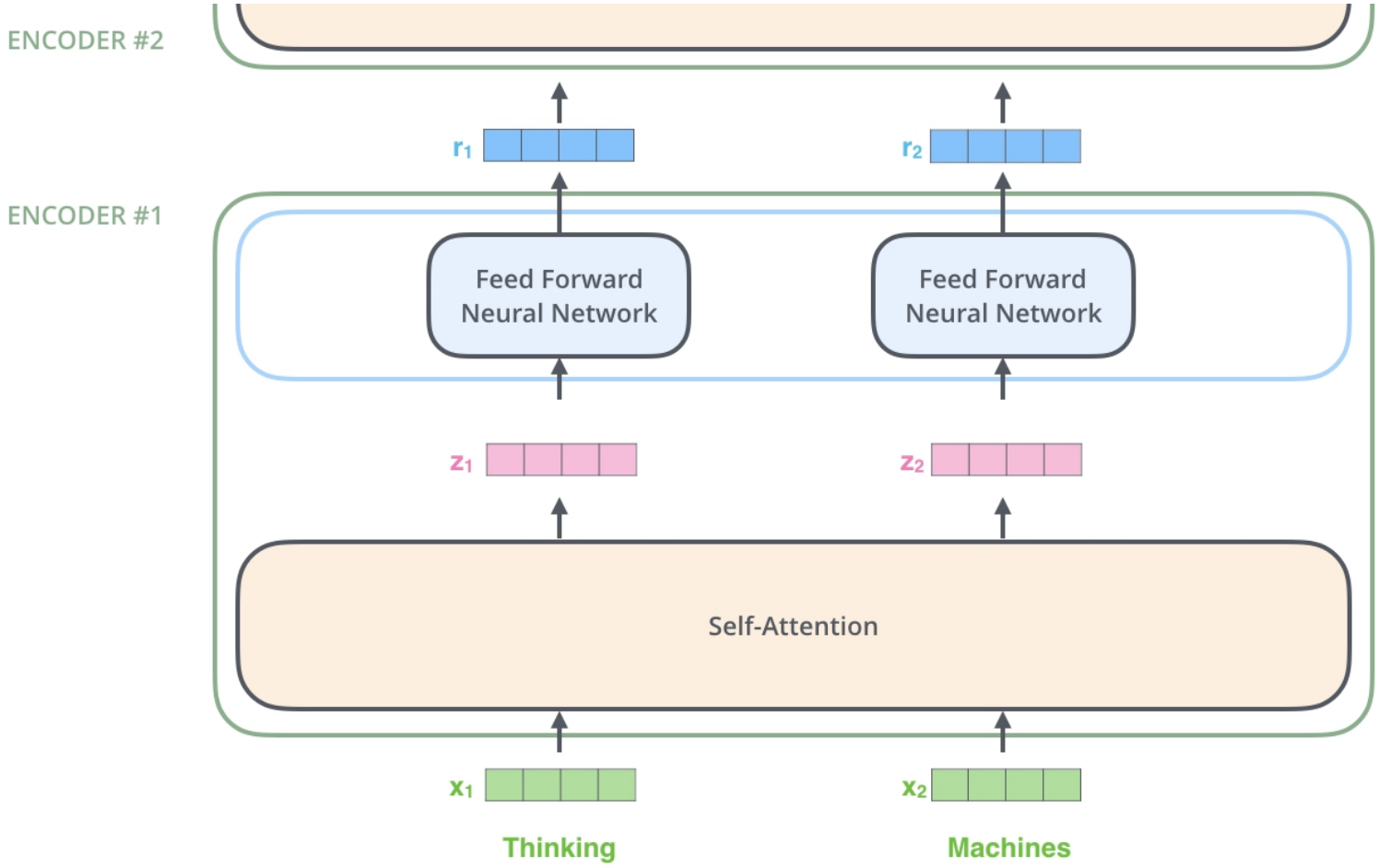


Parallel

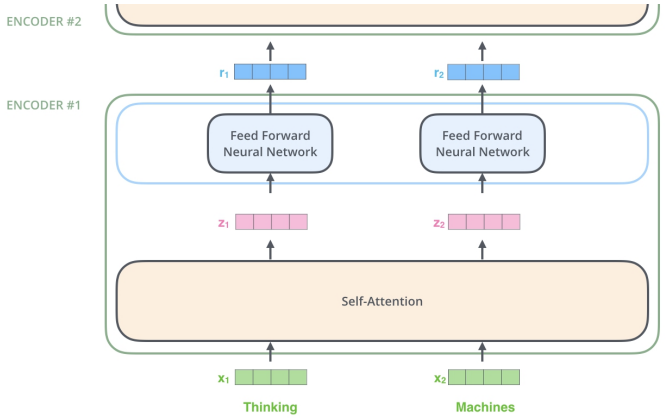
Let’s encoding!

Q:Why feed forward NN?



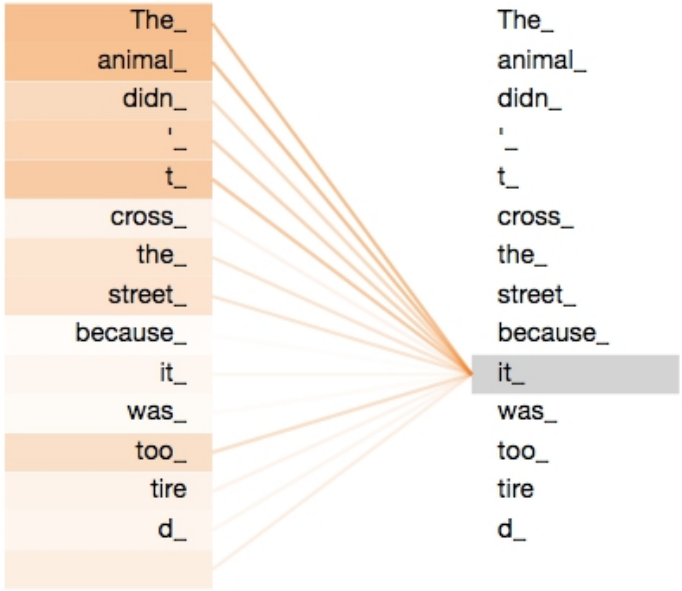


Wait.. What is self-attention?



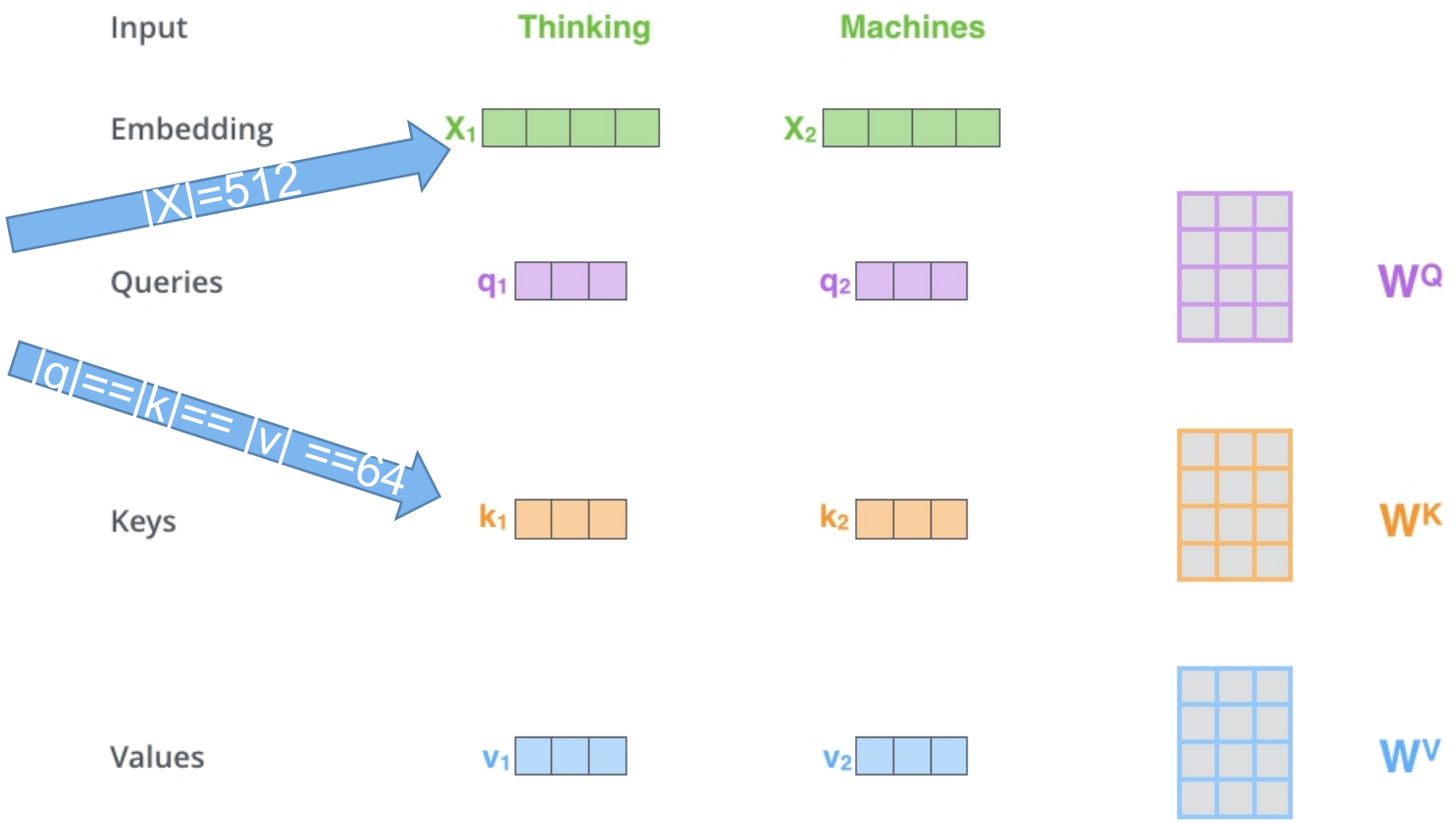
”The animal didn't cross the street because **it** was too tired”

Animal? **Street?**



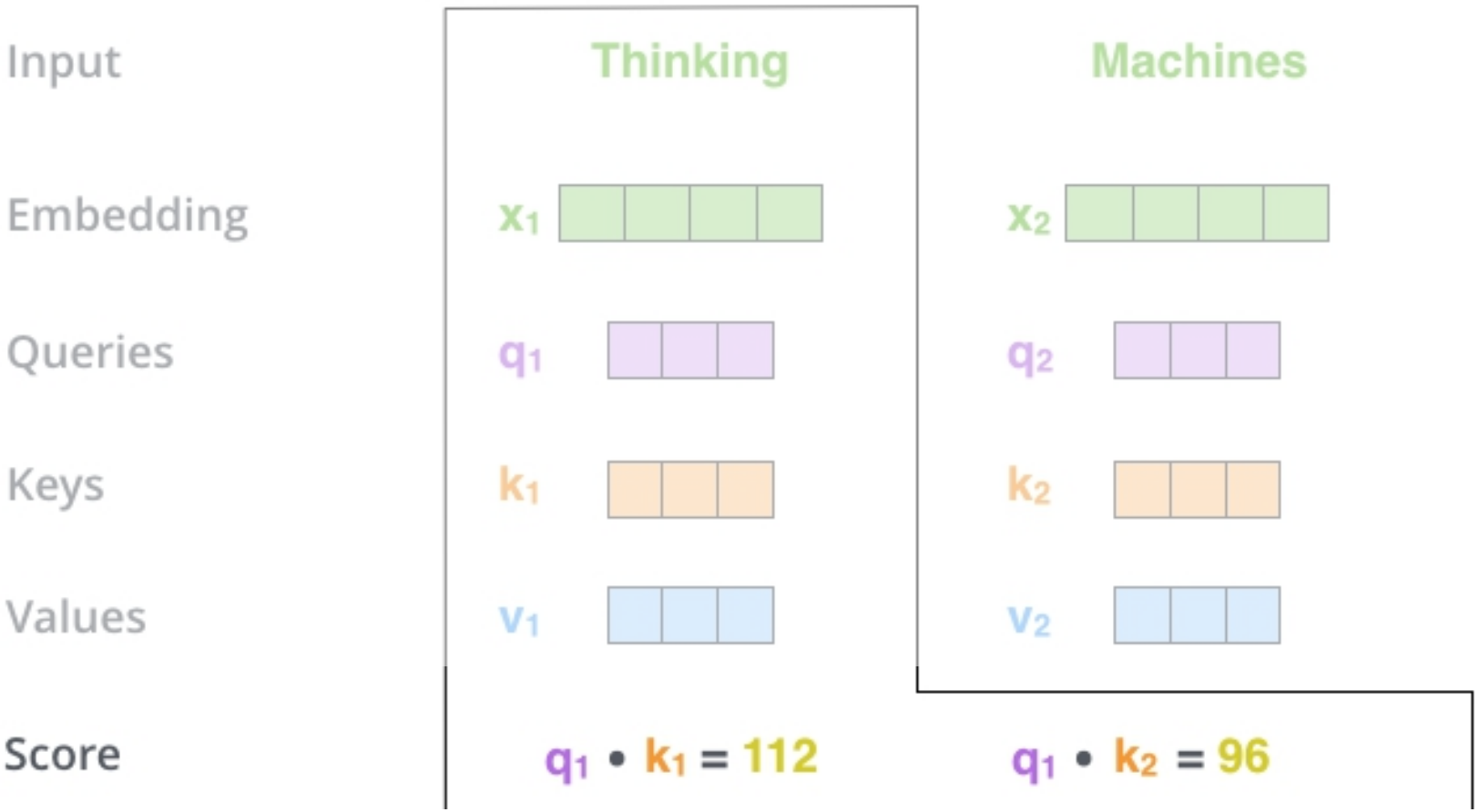
Okay.. How to calculate self-attention?

Intuition: make the computation of multiheaded attention (mostly) constant.

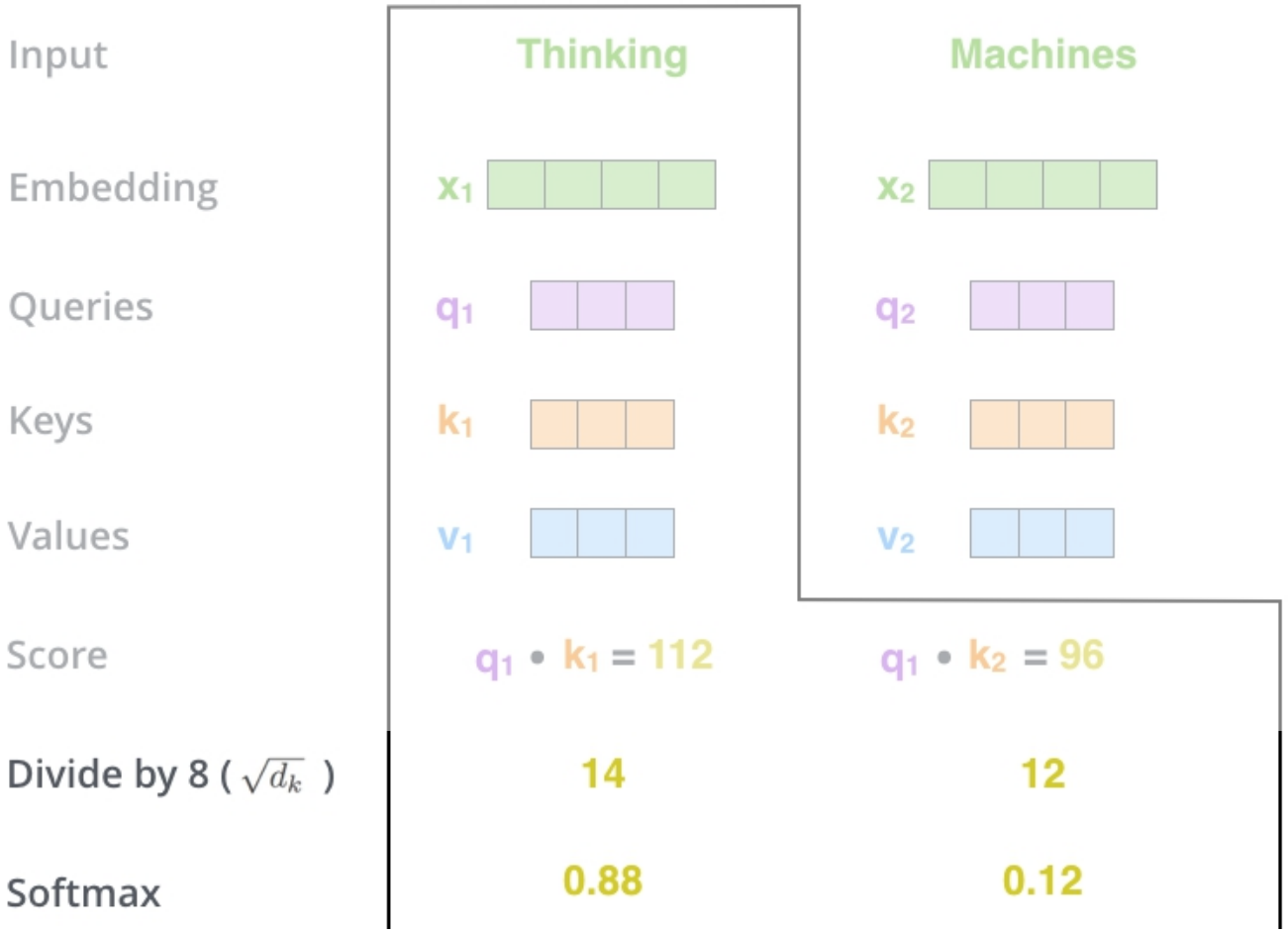


Multiplying x1 by the WQ weight matrix produces q1, the "query" vector associated with that word. We end up creating a "query", a "key", and a "value" projection of each word in the input sentence.

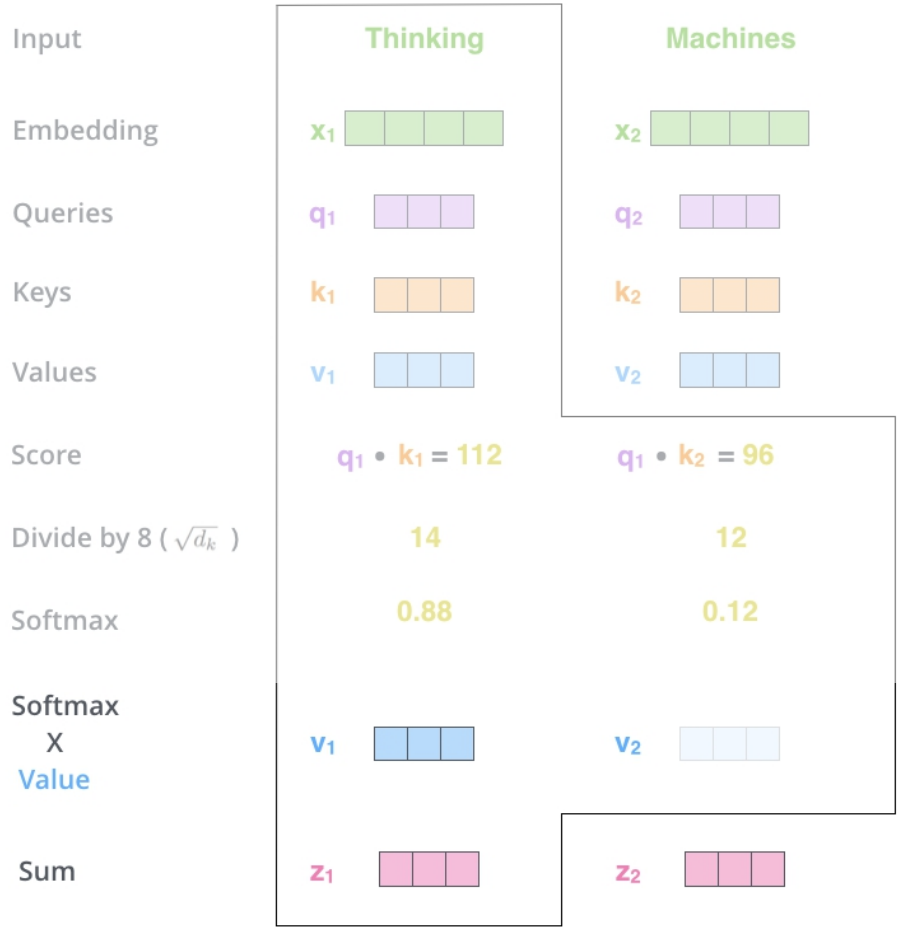
Okay.. How to calculate self-attention?



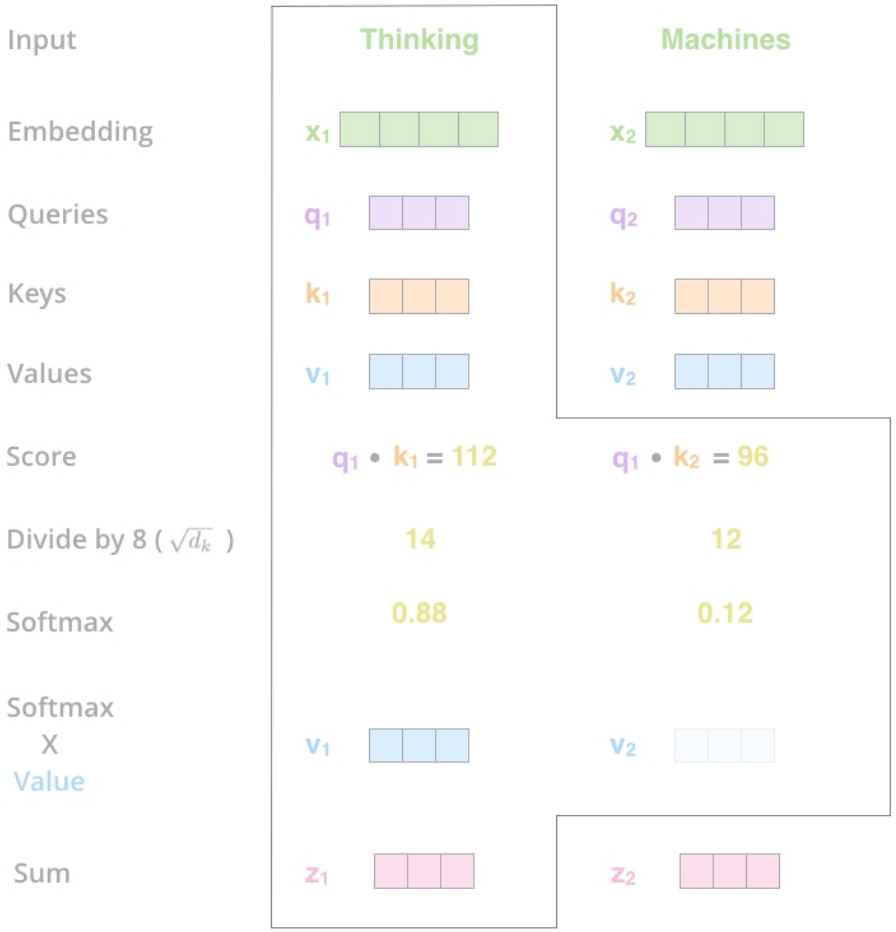
Okay.. How to calculate self-attention?

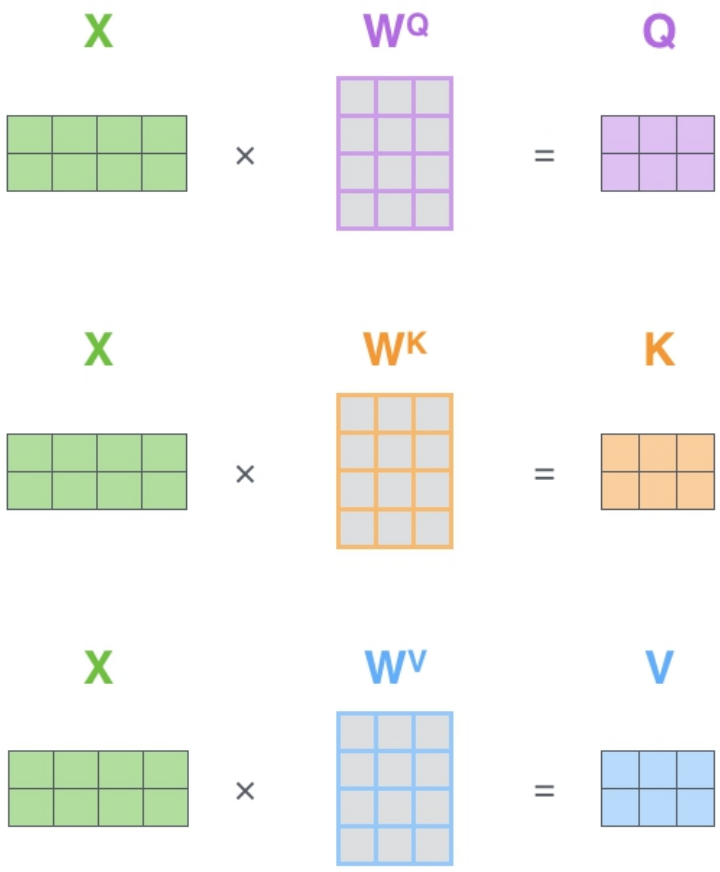


Okay.. How to calculate self-attention?

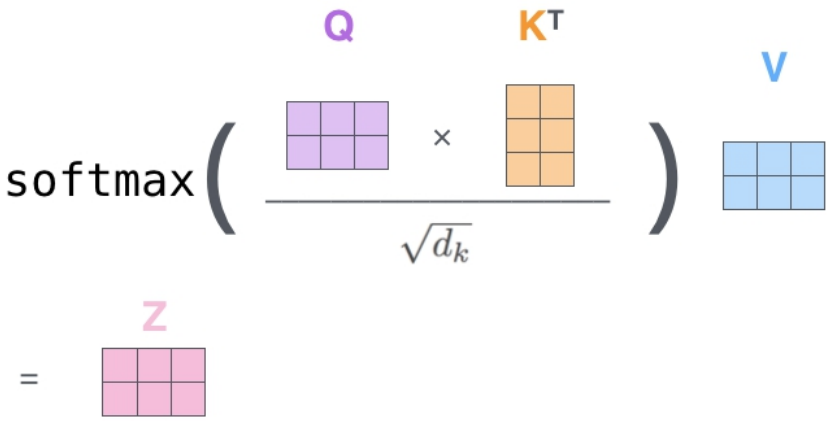


Okay.. How to calculate self-attention?



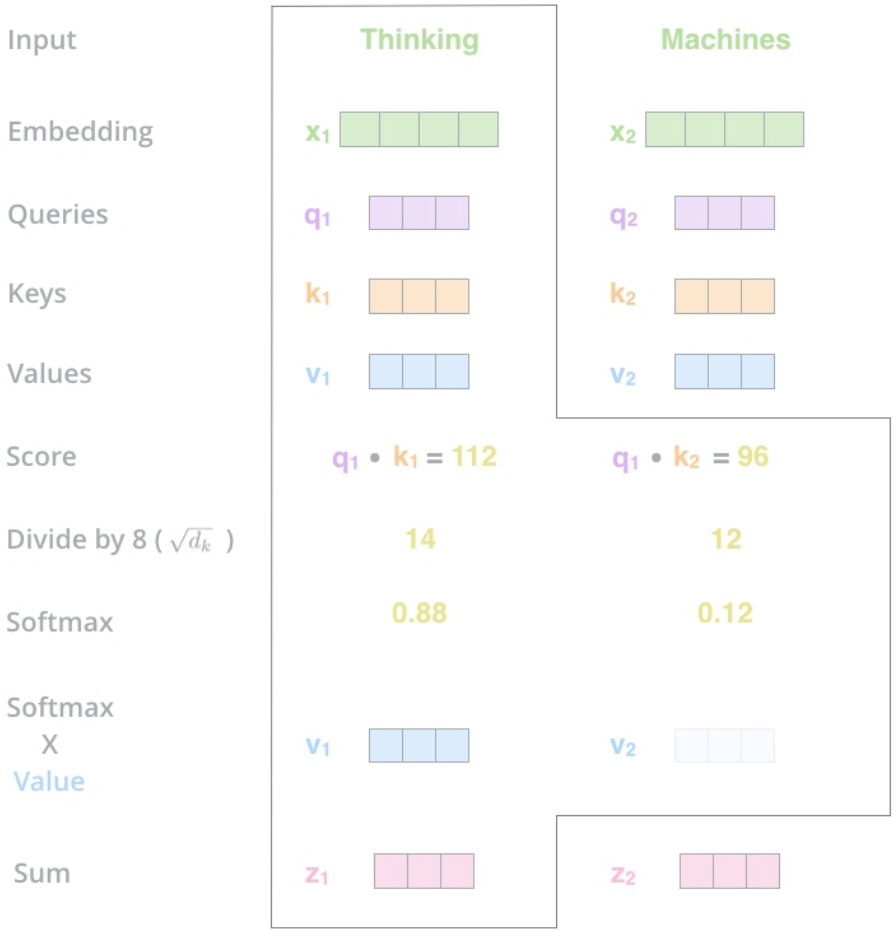


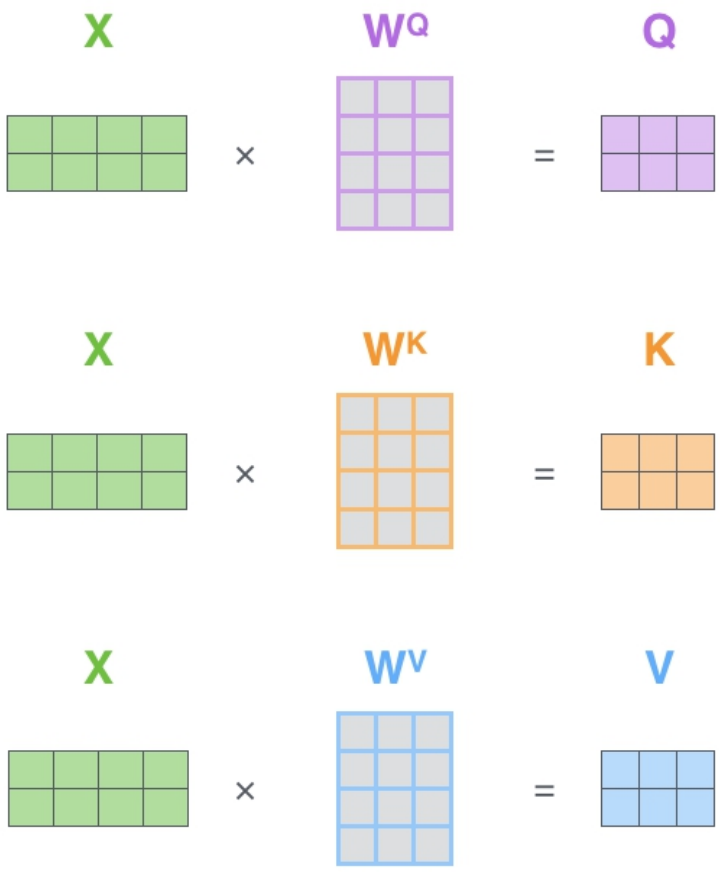
Okay.. How to calculate self-attention?



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

Okay.. How to calculate self-attention?





Okay.. But why? What is **Q,K,V**?

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

- In fact, self-attention calculation is this, why?

$$Softmax(XX^T)X$$

- Dot product of two vectors a means the projection

$$\begin{aligned}[1, 3, -5] \cdot [4, -2, -1] &= (1)(4) + (3)(-2) + (-5)(-1) \\ &= 4 - 6 + 5 \\ &= 3\end{aligned}$$

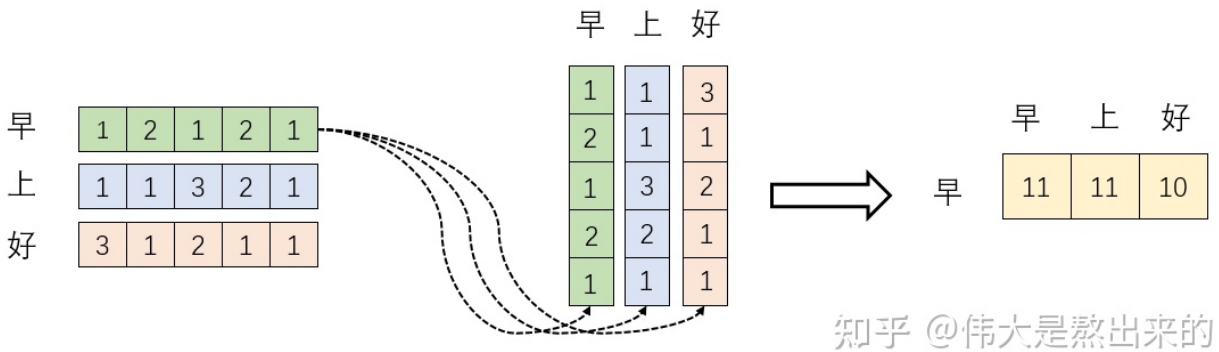
- The projection of vector a into vector b/The dot product of measures} the **similarity** of two vectors.

Okay.. But why? What is **Q,K,V**?

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

- In fact, self-attention calculation is this, why?

$$Softmax(XX^T)X$$



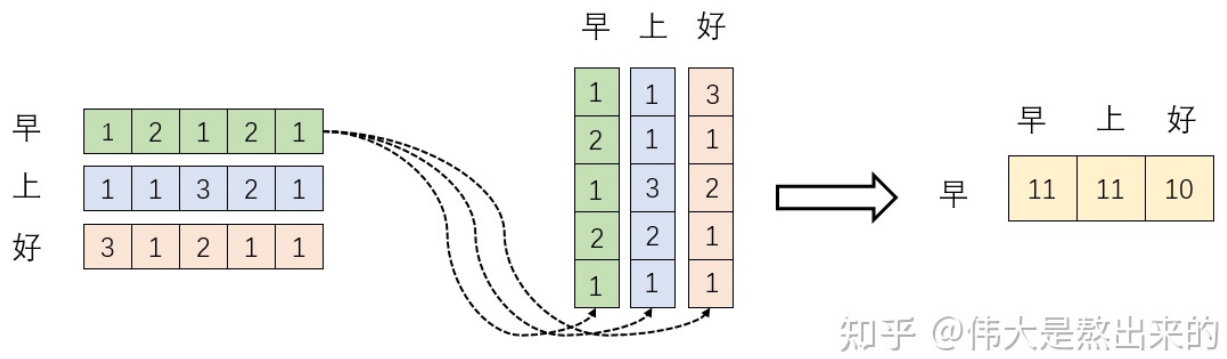
$$Softmax(z_i) = \frac{e^{z_i}}{\sum_{c=1}^{C} e^{z_c}}$$

Okay.. But why? What is **Q,K,V**?

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

- In fact, self-attention calculation is this, why?

$$Softmax(XX^T)X$$

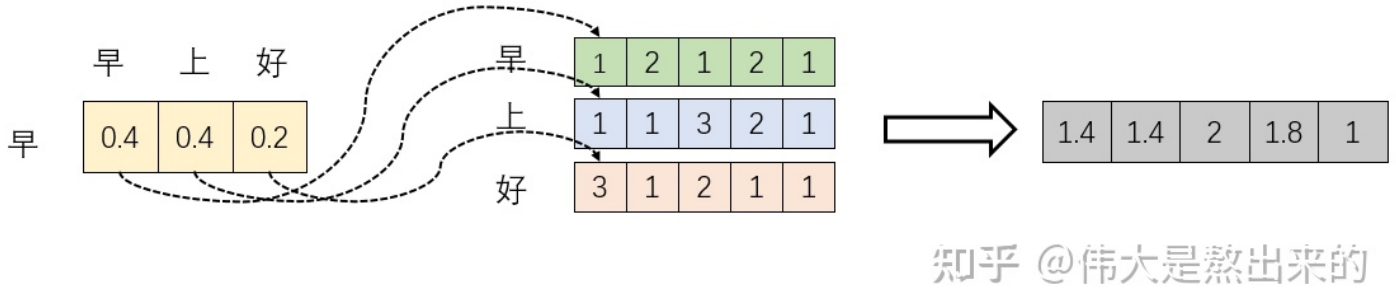


Okay.. But why? What is **Q,K,V**?

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

- In fact, self-attention calculation is this, why?

$$Softmax(XX^T)X$$

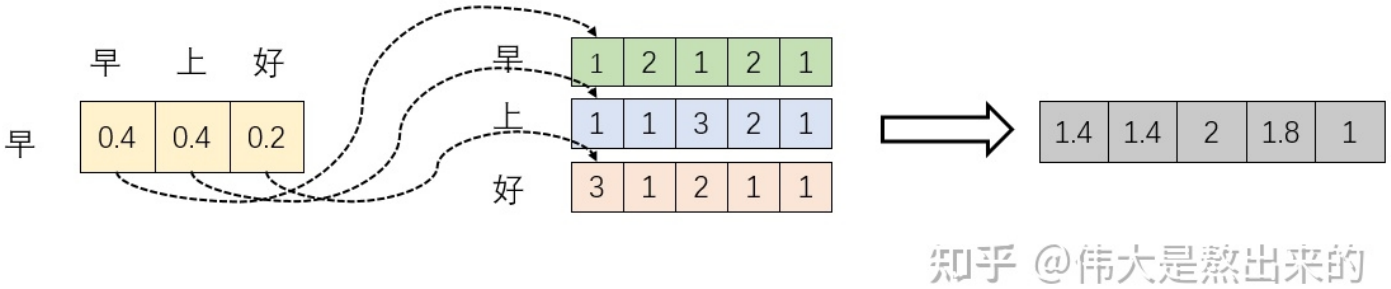


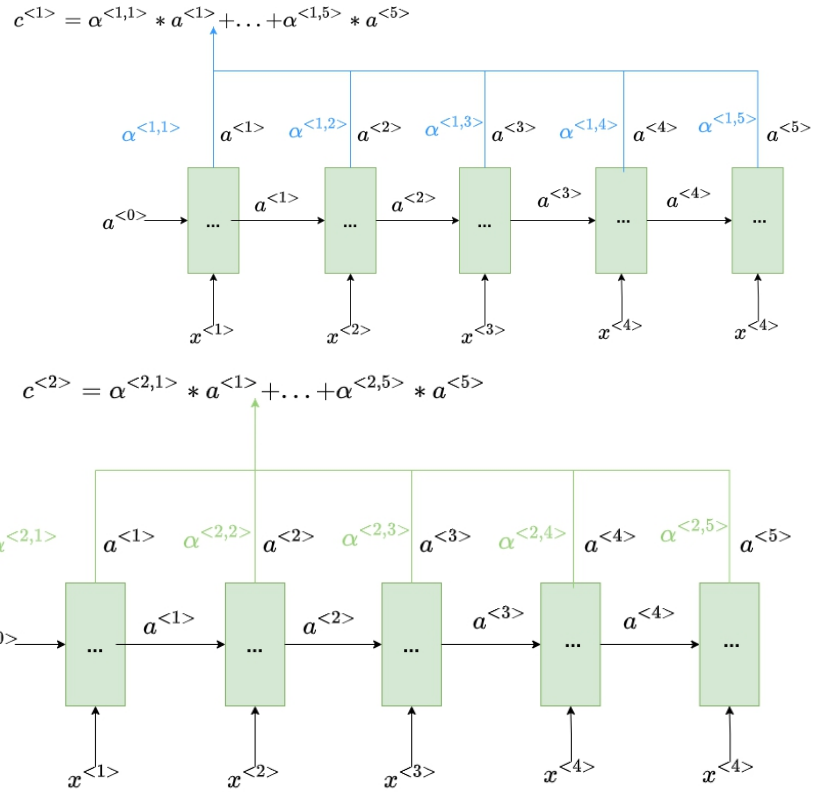
Okay.. But why? What is **Q,K,V**?

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

- In fact, self-attention calculation is this, why?

$$Softmax(XX^T)X$$

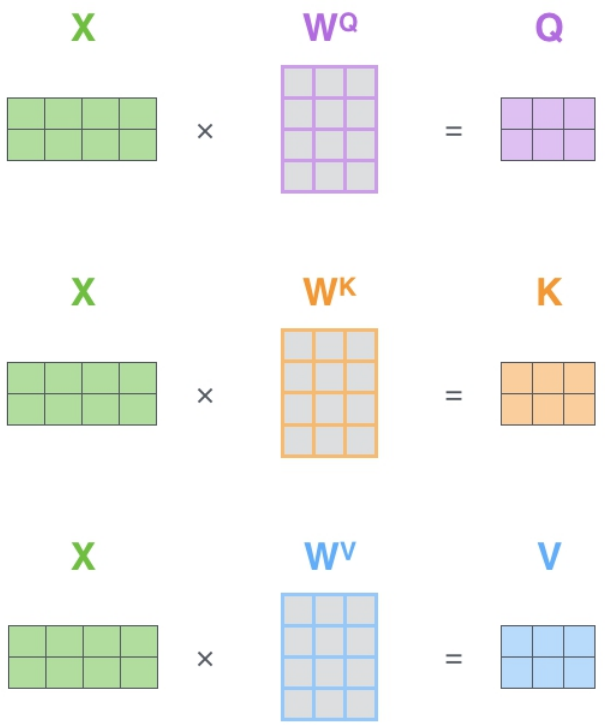




How does **Q,K,V** correlates with X?

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

$$Softmax(XX^T)X$$

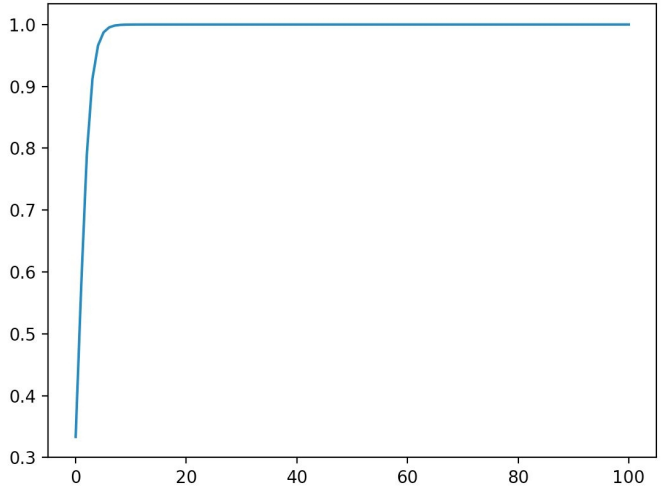


What about $\sqrt{d_k}$?

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

Addictive attention computes the compatibility function using a feed-forward network with a single hidden layer. Dot product attention is much faster and space-efficient.

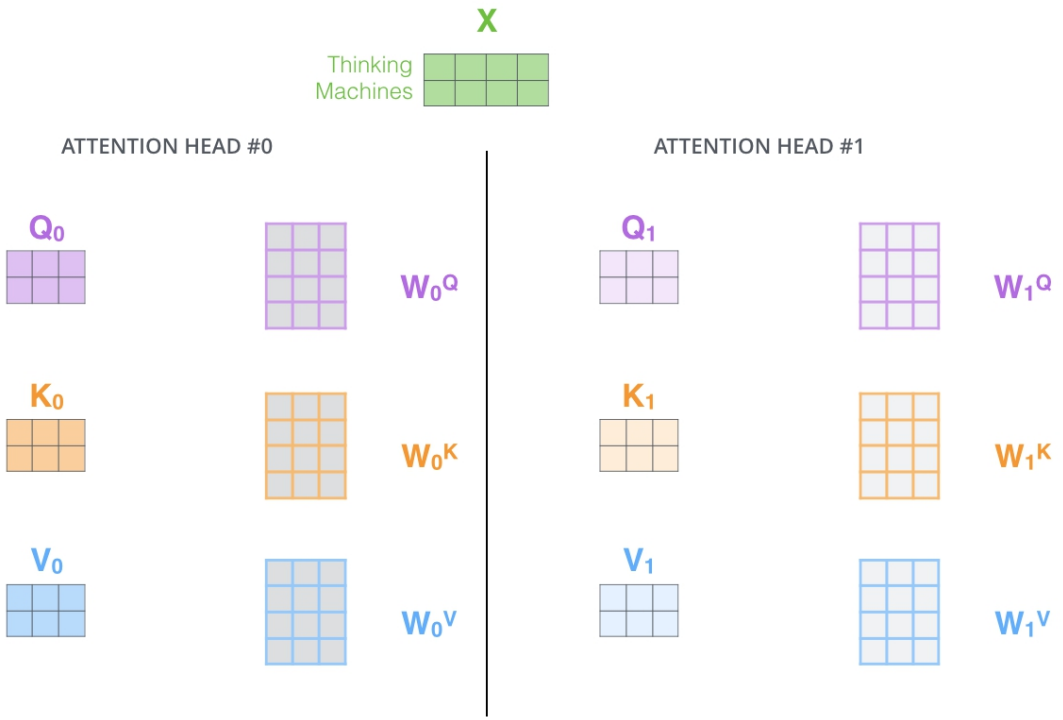
For large values of dk will also make dot-product attention large.

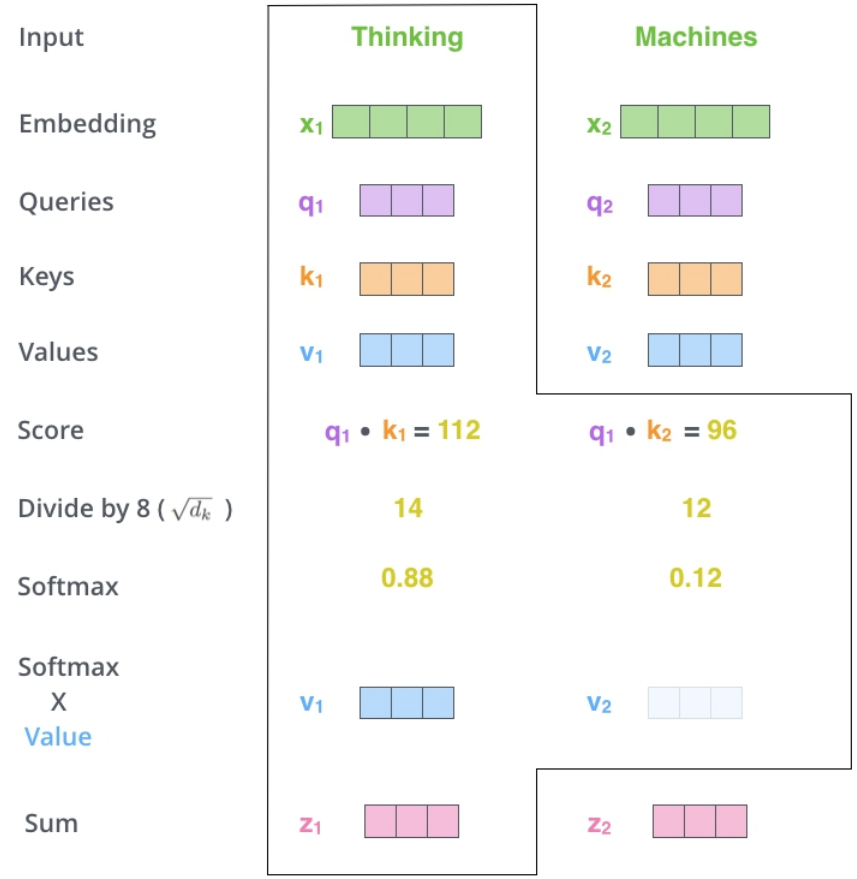


$$Softmax(z_i) = \frac{e^{z_i}}{\sum_{c=1}^{C} e^{z_c}}$$

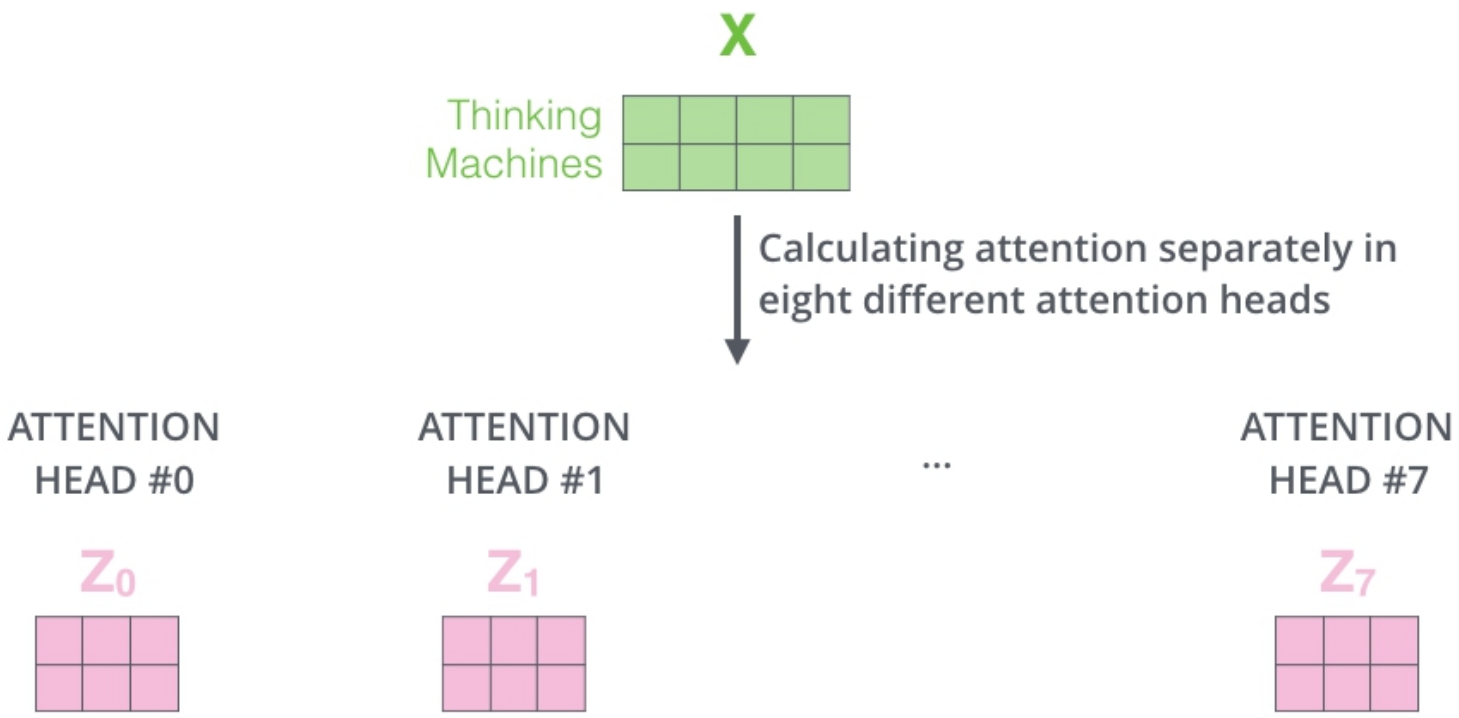
Proof: https://www.zhihu.com/question/339723385 (If you need further explaination, email me.)

What is multi-head self-attention





What is multi-head self-attention



What is multi-head self-attention

1) This is our input sentence*

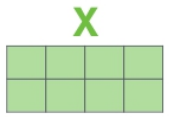
2) We embed each word*

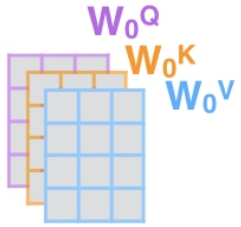
3) Split into 8 heads. We multiply X or R with weight matrices

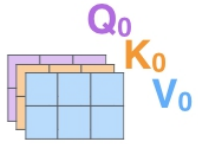
4) Calculate attention using the resulting Q/K/V matrices

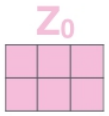
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer

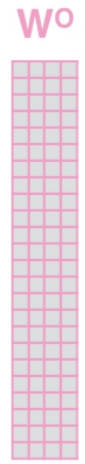
Thinking
Machines

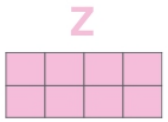




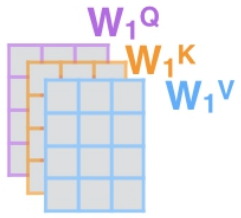


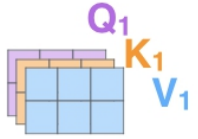


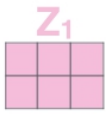


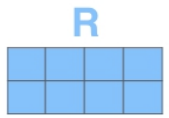


* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one

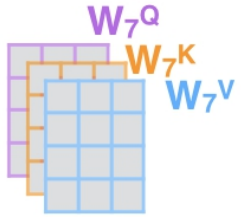




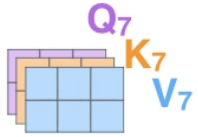




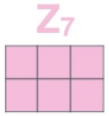
...



...

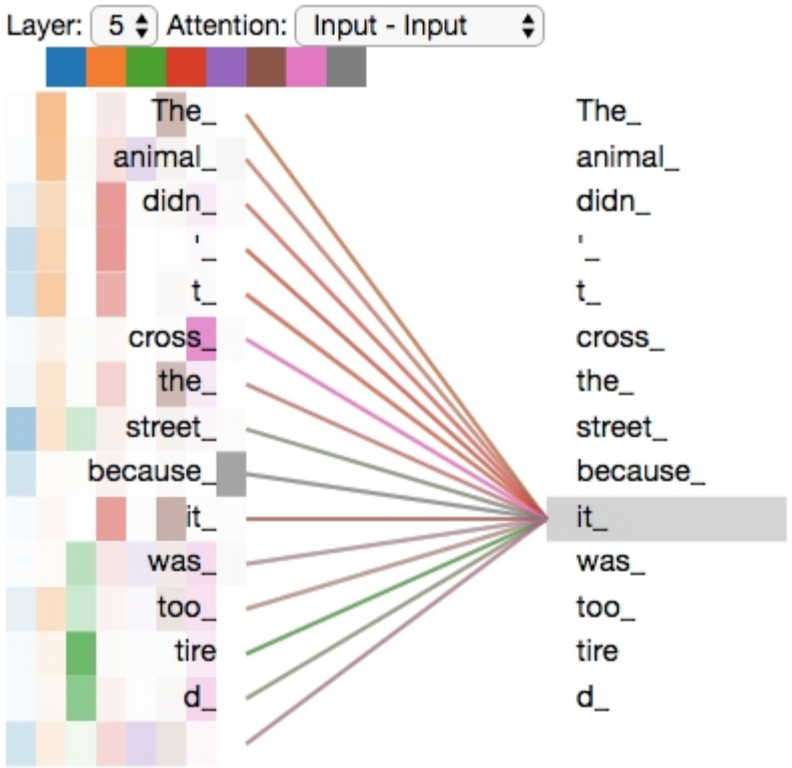


...



Why **multi-head** self-attention

- It expands the model's ability to focus on different positions.
- It gives the attention layer multiple "representation subspaces". After training, each Q,K,V is used to project the input embeddings (or vectors from lower encoders/decoders) into a different representation subspace.



Decoder side

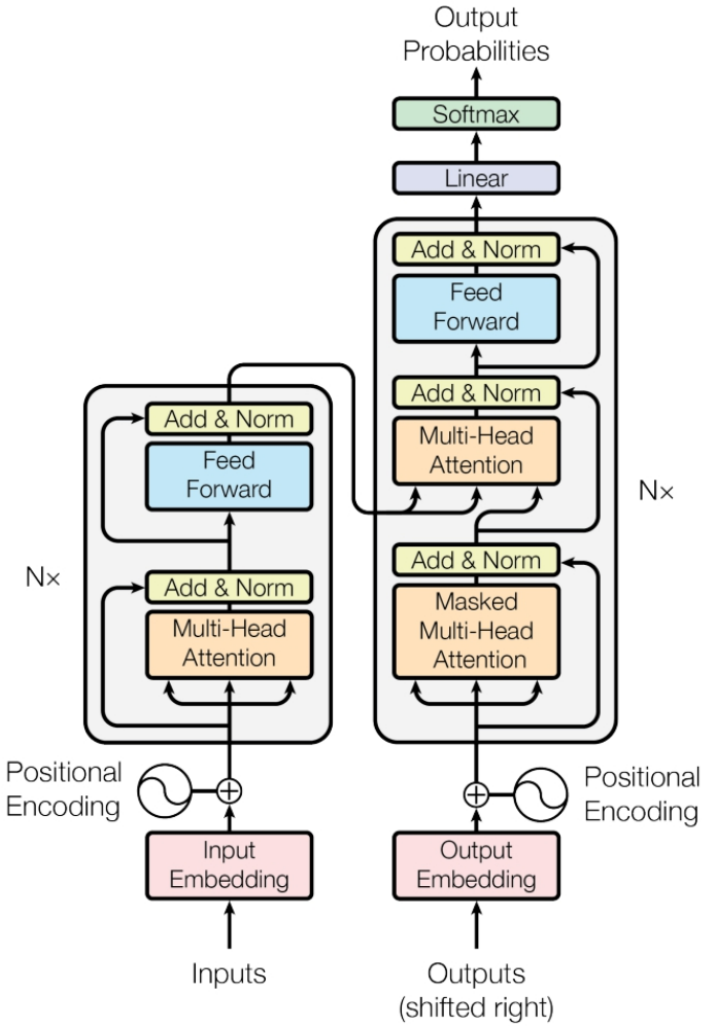
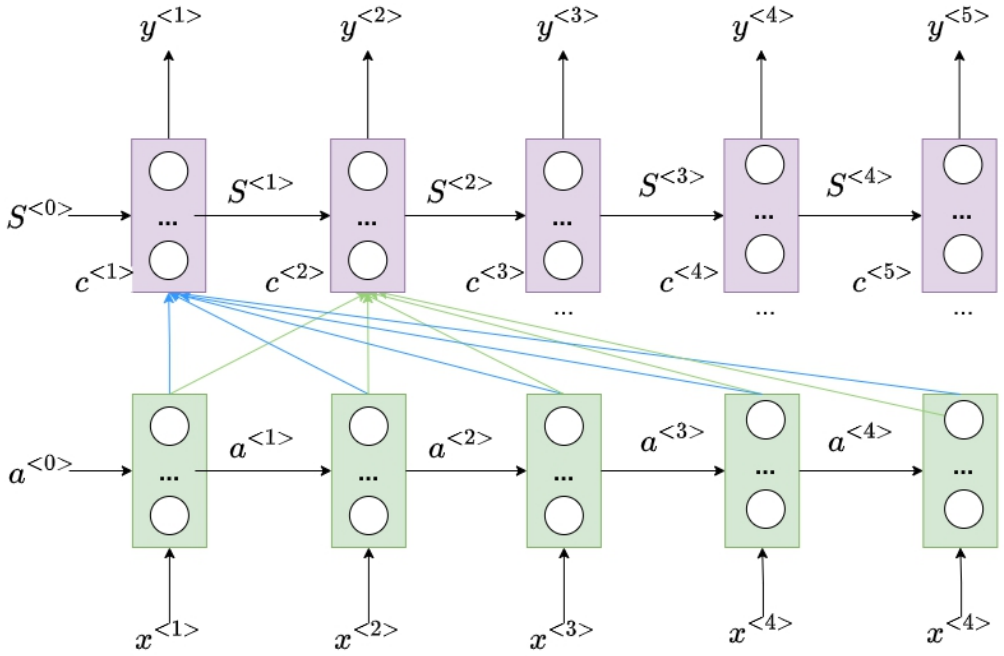
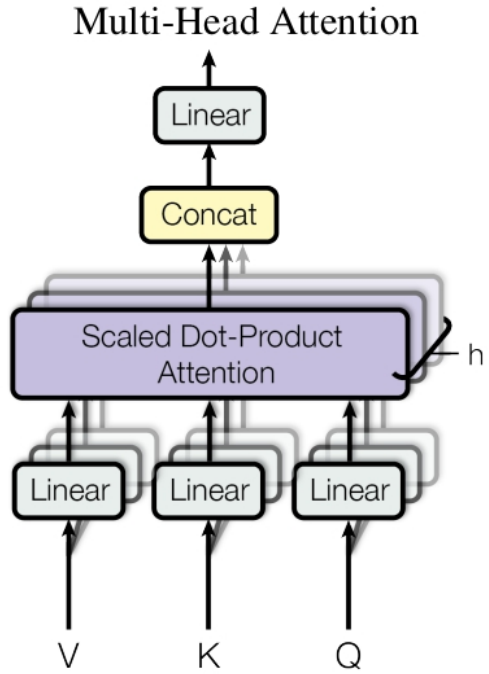


Figure 1: The Transformer - model architecture.

The “Encoder-Decoder Attention” layer works just like multiheaded self-attention, except **it creates its Queries matrix from the layer below it, and takes the Keys and Values matrix from the output of the encoder stack.**

Positional encoding





- Position information for words in sequence is lost.
- But postition information is necessary to help translation.
- **Idea: Try to encode position information inside the input embedding.**

Positional encoding

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

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

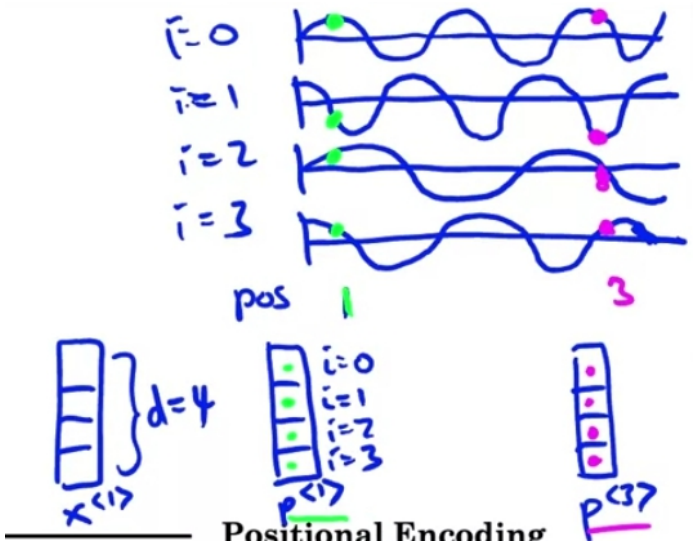
Why choose this encoding?

- For any fixed offset k, PE_{pos+k} can be represented as a linear function of PE_{pos}

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

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

- i: encoding dim
- pos: word posi



Masking

- **Intuition: Let the model only focus on what it should focus on.**
- Padding mask: (Set “padded element attention to -inf”)

- Sequence mask: Bans the decode from seeing future information. (Set “future attention it -inf”)

Residual

- **Intuition: When the model goes deeper, gradient vanishing, gradient expolosion, and saturation may occur.**

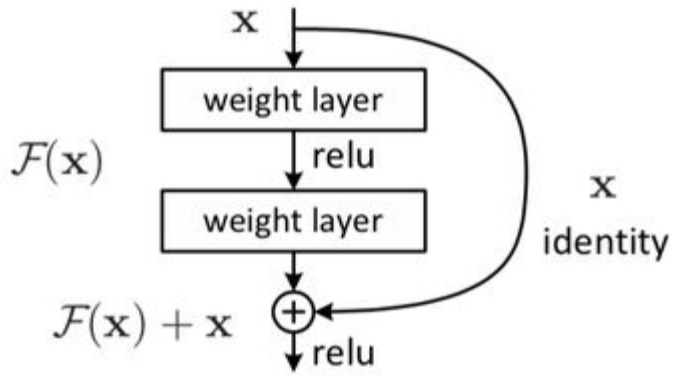


Figure 2. Residual learning: a building block.

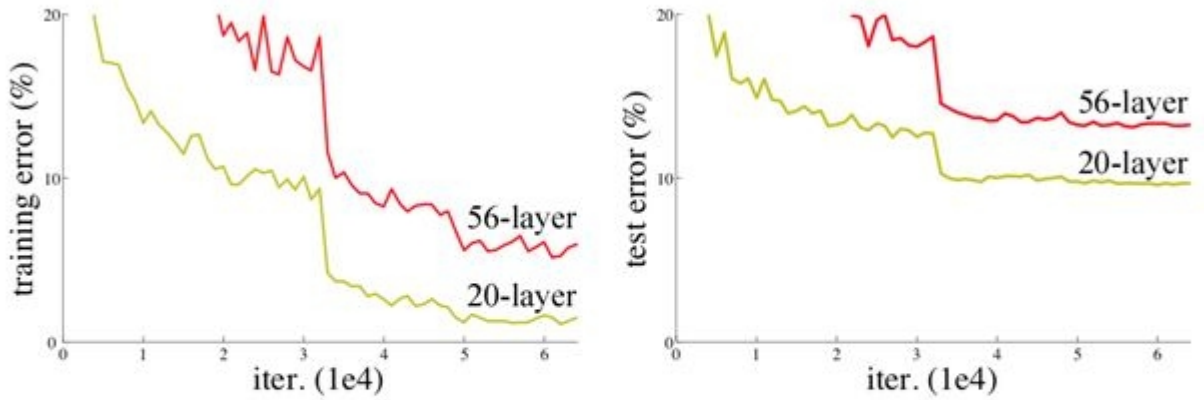


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

Training

- Encoder-decoder is trained by forward-backward propagation (Just like RNN).
- The result is P(x=current word|previous sentences). Beam search is used to generate meaningful sencentes.