教材《机器学习与大语言模型》配套电子教案

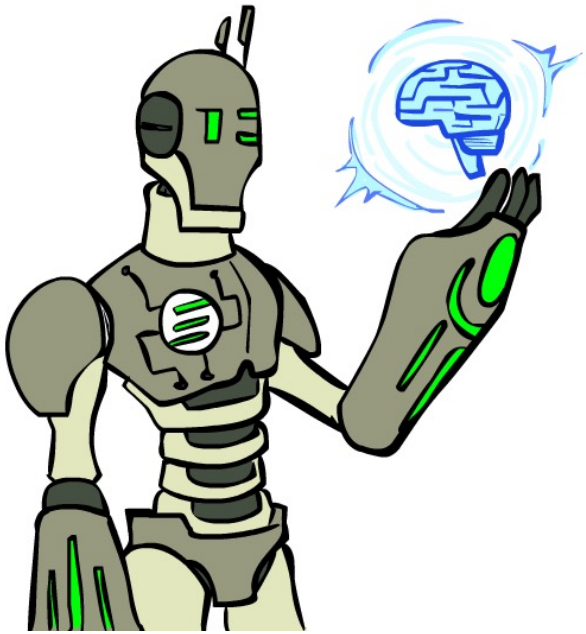
机器学习

第十讲:Transformer

顾小东
上海交通大学计算机学院

Today

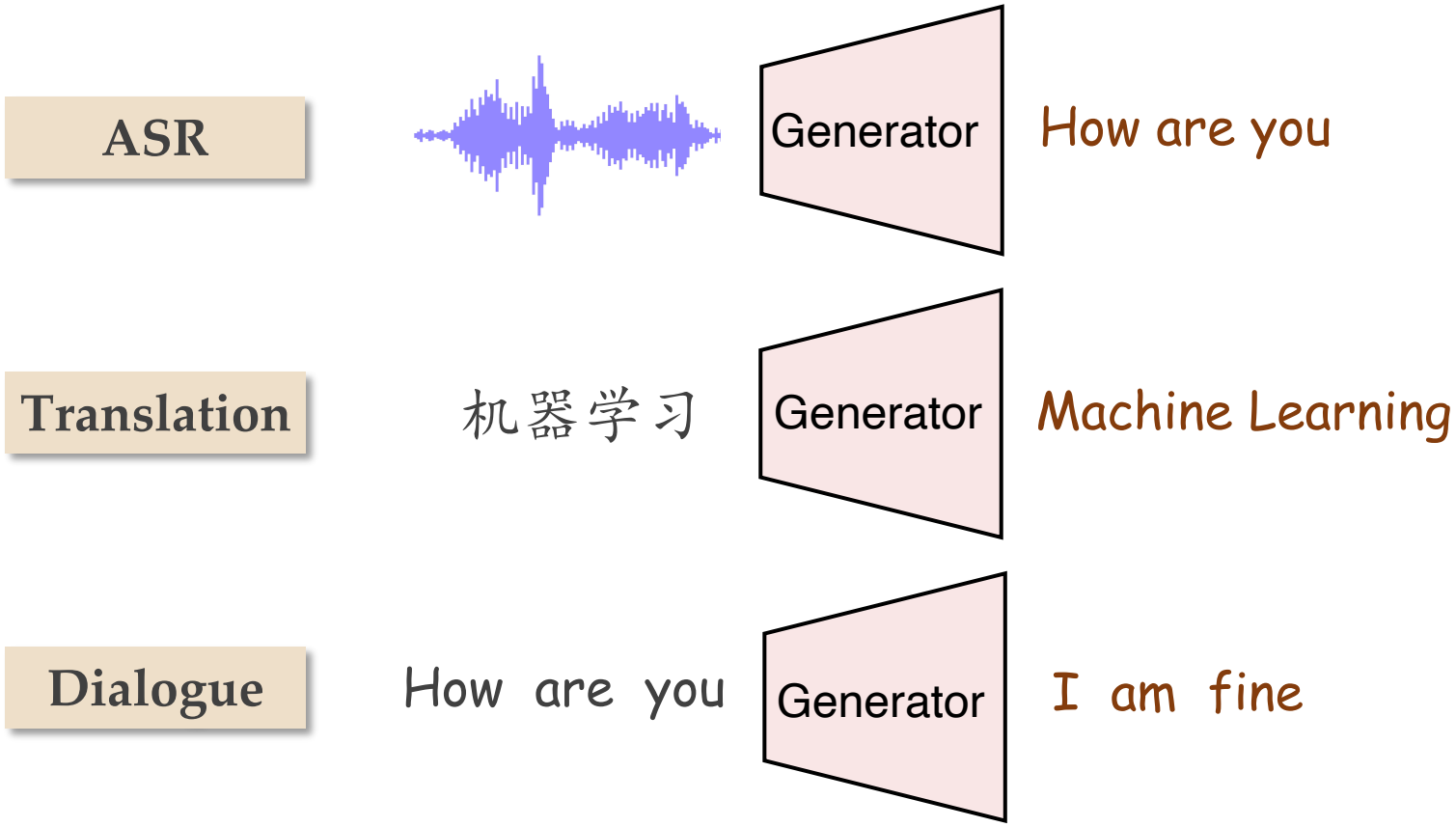
- RNN Encoder-Decoder
- Attention
- Transformer



Sequence-to-Sequence Learning

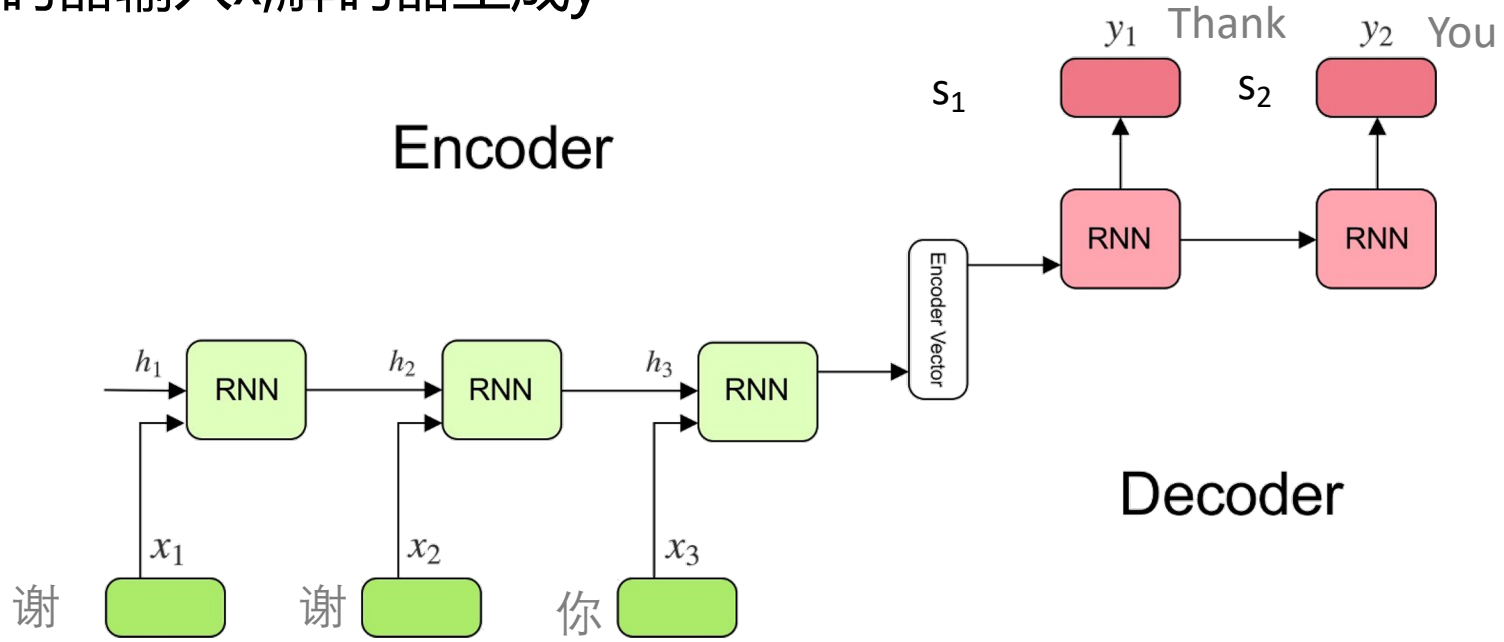
条件式序列生成任务

- 在一个(源)序列的条件下，生成另一个(**目标**)序列.



RNN编解码器模型 (Cho 2014)

- 输入序列$x = (x_1, \cdots, x_{T_x})$, 生成序列$y = (y_1, \cdots, y_{T_y})$
- 采用两个RNN,分别作为编码器(Encoder)和解码器(Decoder), 编码器输入x,解码器生成y



- 也被称为**序列到序列学习**(Sequence-to-Sequence Learning)

模型训练

数据: $D = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})\}$,

其中 $x^{(\ell)} = (x_1, \ldots x_{Tx})$, $y^{(\ell)} = (y_1, \ldots y_{Ty})$.

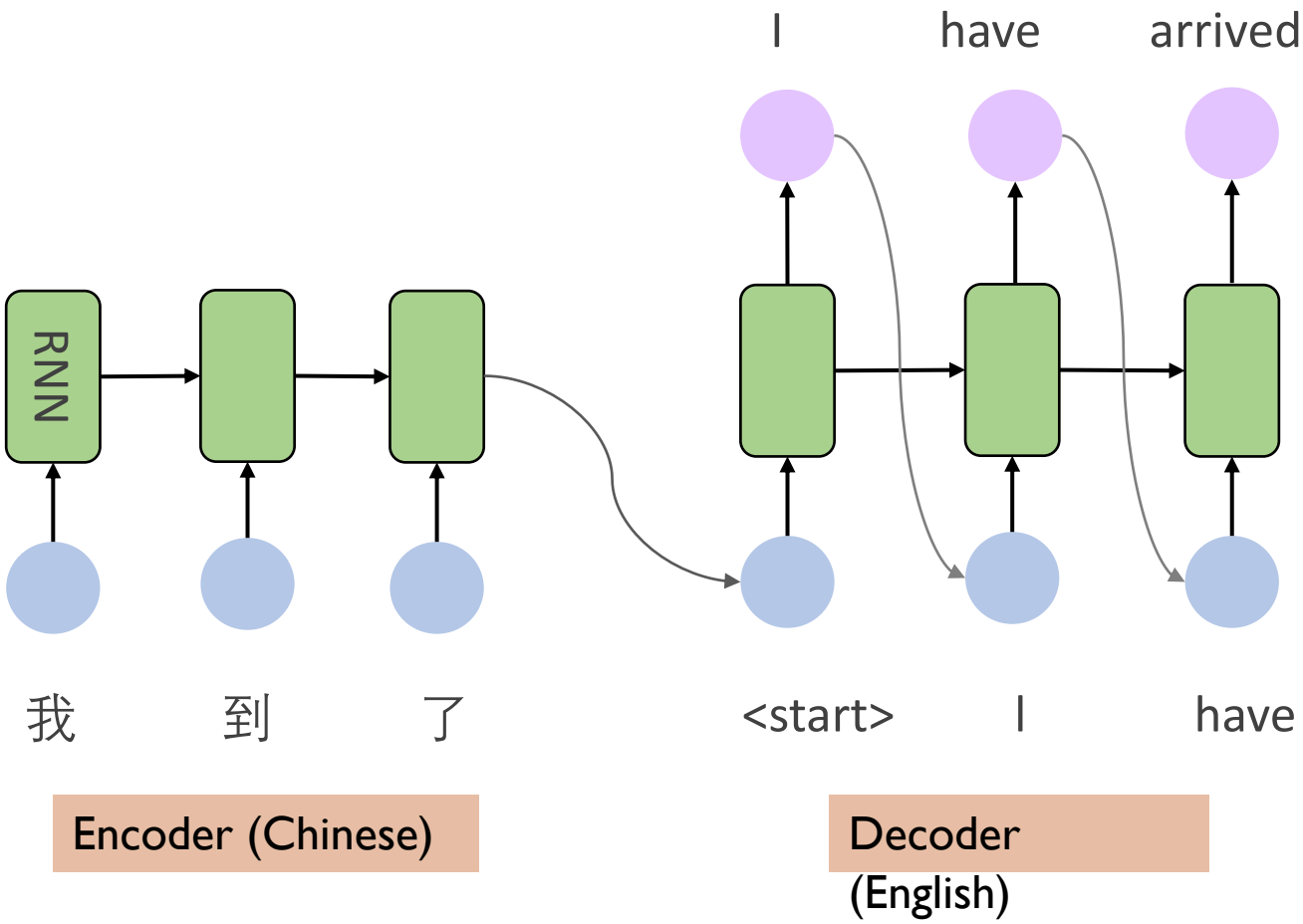
损失函数 – 最小化交叉熵:

$$l(\theta|x, y) = -\sum_{t=1}^{T} \log p_\theta (y_t | y_1, \ldots, y_{t-1}, x)$$

$$L(\theta|D) = -\frac{1}{N} \sum_{l=1}^{N} l(\theta|x^{(l)}, y^{(l)})$$

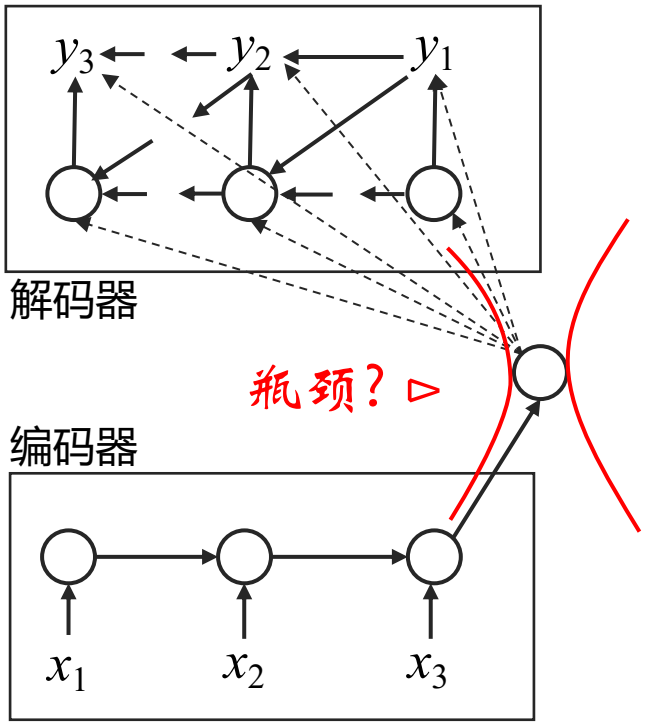
优化算法 – 梯度下降

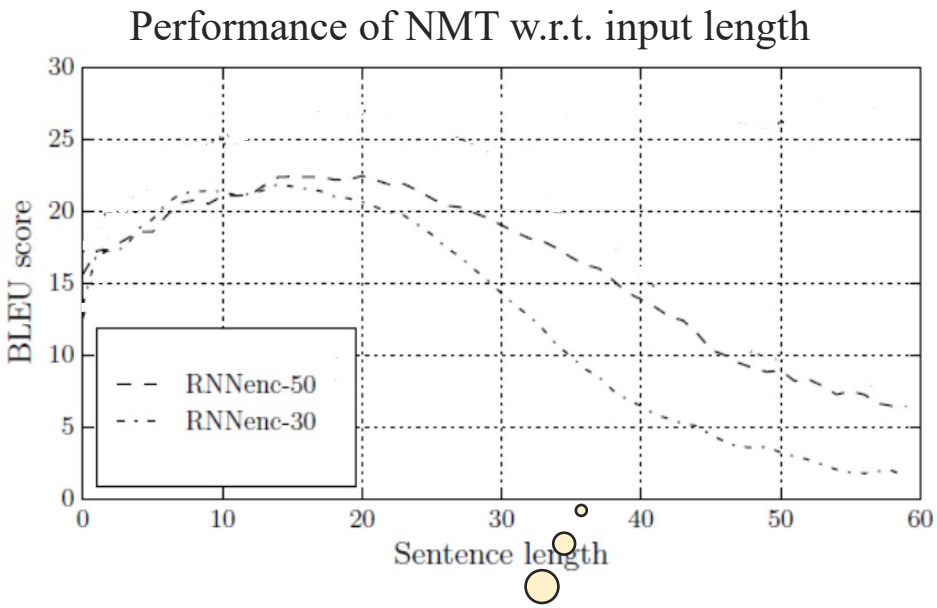
举例：机器翻译


I
have
arrived
RNN
我
到
了
<start>
I
have
Encoder (Chinese)
Decoder
(English)

长序列的瓶颈

核心问题: 一个超长的序列被压缩成一个向量，没有考虑到序列中每个位置的重要性。





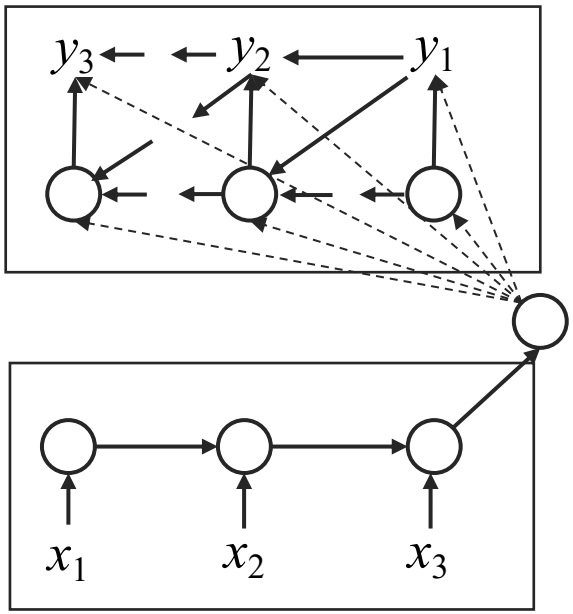
实验观察：模型性能随着序列长度增大而降低

基于注意力的Sequence-to-Sequence学习

No Attention

根据单一、固定的编码向量解码

$h_i = \text{RNN}_{\text{out}}(h_{i-1}, y_{i-1}, \boldsymbol{c})$

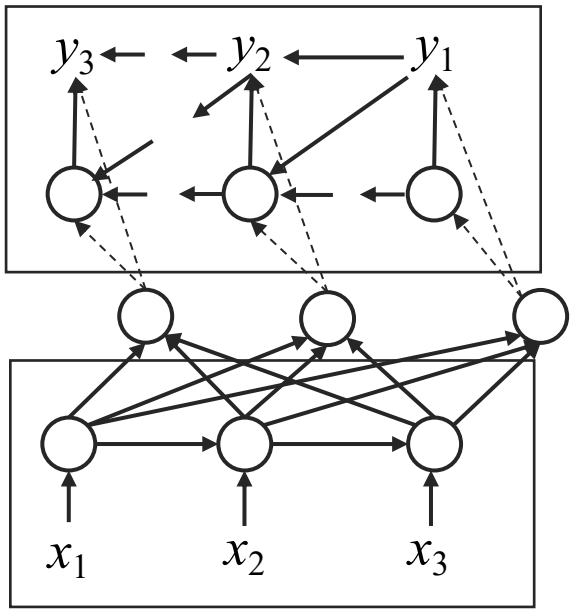


VS

With Attention

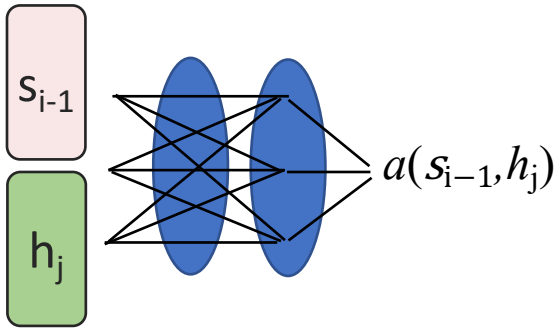
在上下文中动态选择感兴趣的内容组成编码向量。

$h_i = \text{RNN}_{\text{out}}(h_{i-1}, y_{i-1}, \boldsymbol{c_i})$



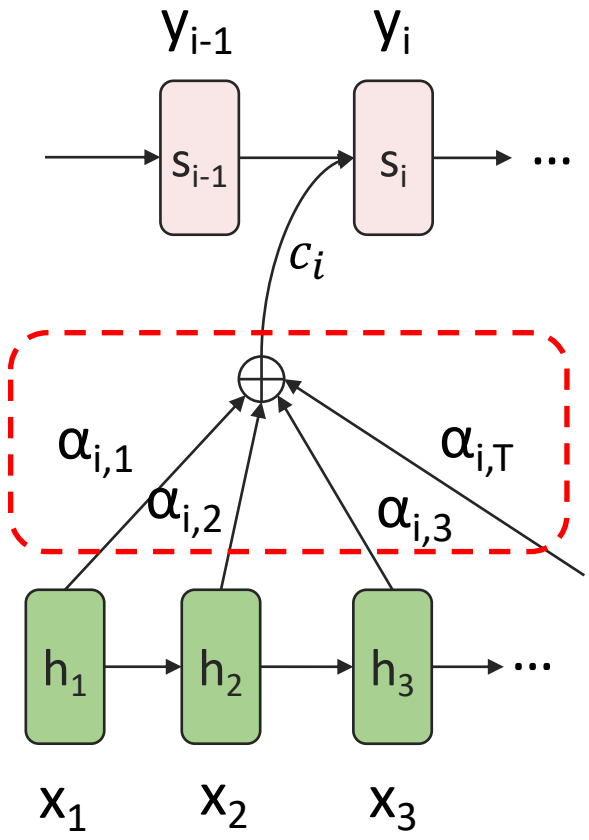
注意力模型

- 令$a_{i,j}$为注意力分数, 代表编码器每个输入状态 $\mathbf{h}_j$ 和解码器每个输出状态s_i的匹配程度
- 不仿用一个神经网络来估算该分数:
  e.g., $a_{i,j}=NN(\boldsymbol{s}_{i-1}, \boldsymbol{h}_j)$

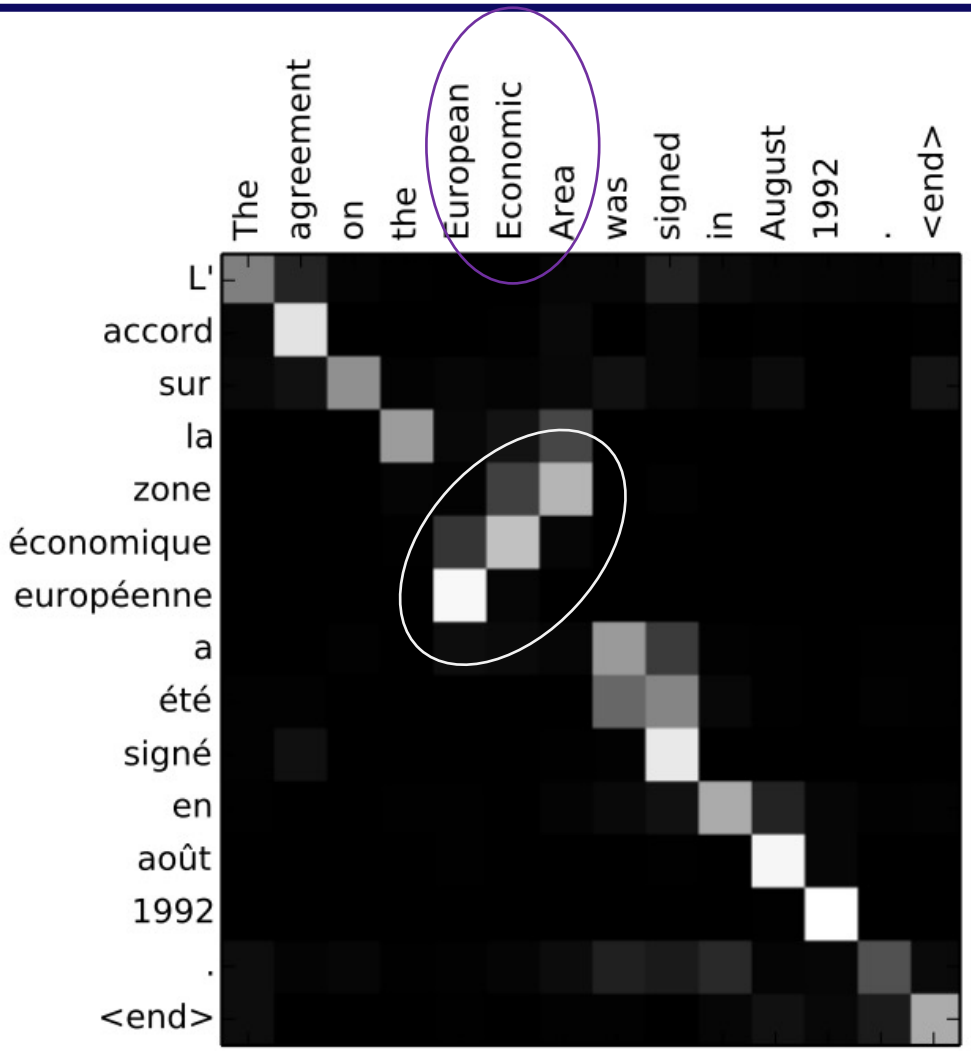


- 进一步通过Softmax归一化为概率:

$$\alpha_i = \text{softmax}(a_{i,1}, a_{i,2,\ldots}, a_{i,T})$$



注意力可视化



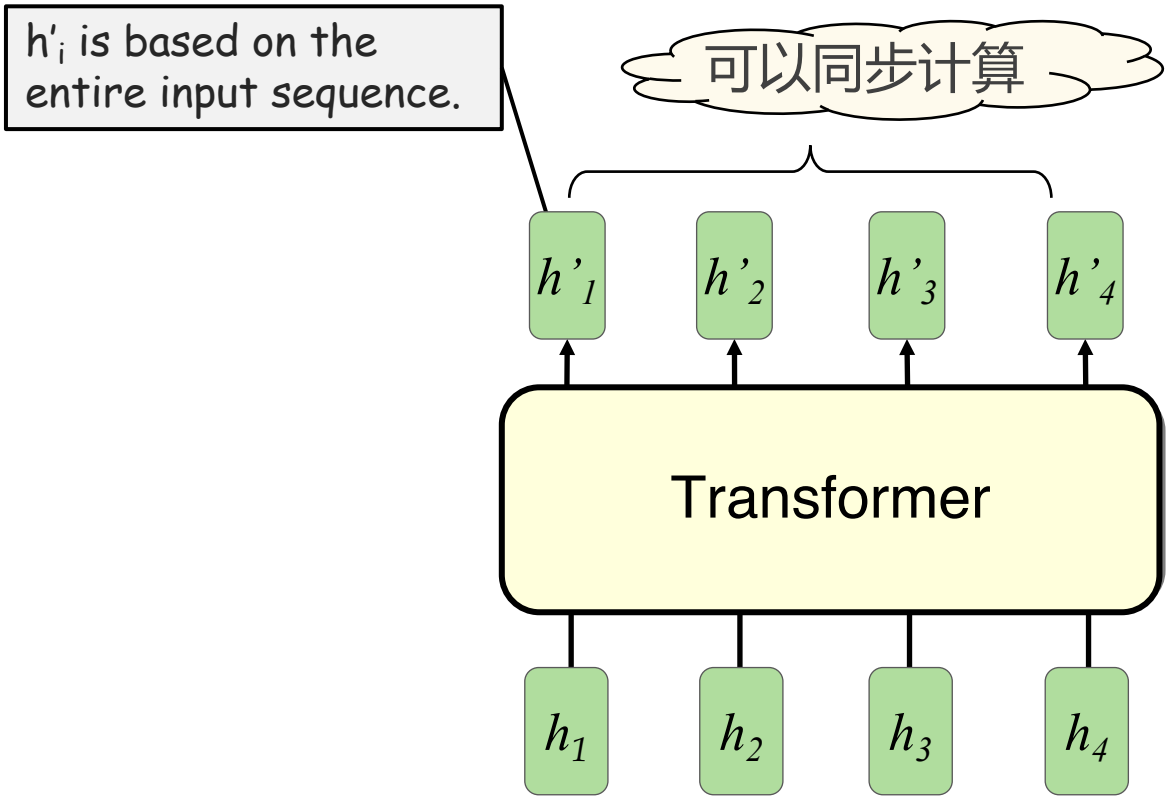
机器翻译模型的真实 α_{ij}

You can see how the model paid attention correctly when outputting "European Economic Area". In French, the **order of these words is reversed** ("européenne économique zone") as compared to English. Every other word in the sentence is in similar order.

Transformer

Seq2seq model with "Self-attention"

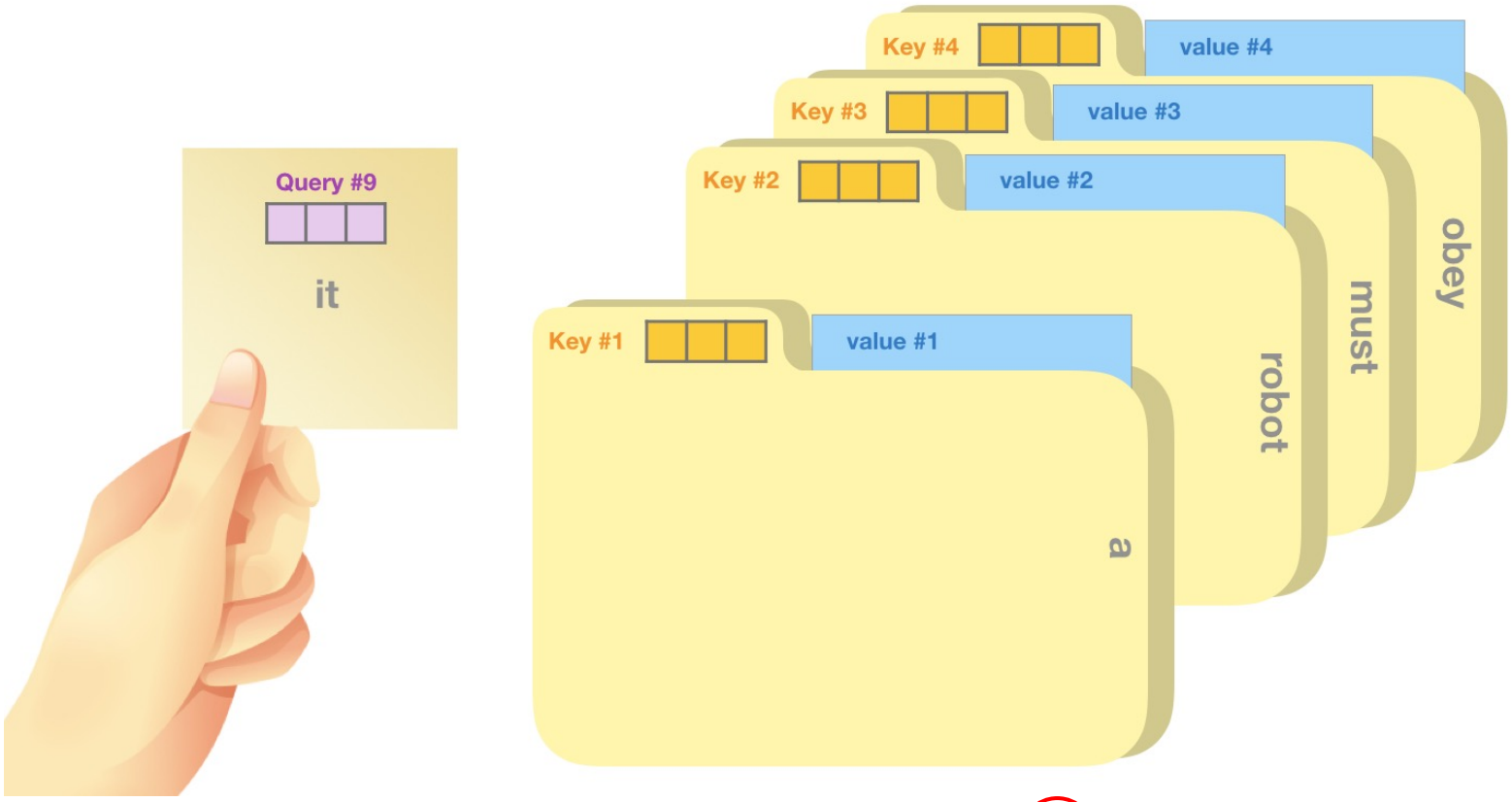
从RNN到自注意力机制



再见RNN: 凡是使用RNN的地方都可以替换为自注意力模型。

自注意力：基本思路

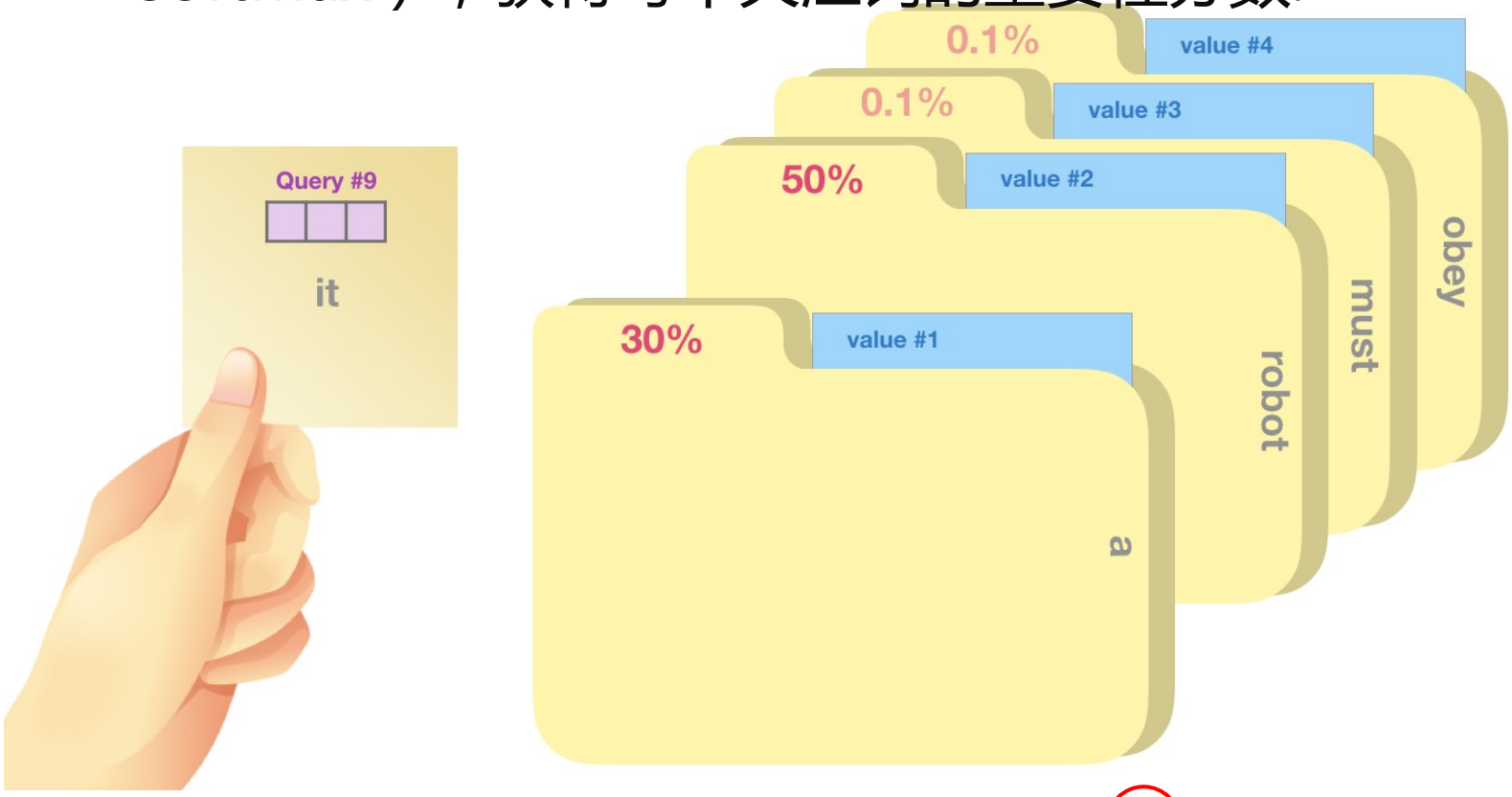
- 让句子中的每个单词都关注其他单词



A robot must obey the orders given it.

自注意力：基本思路

- 将Query向量和每个Key向量进行内积（加softmax），获得每个关注词的重要性分数.



A robot must obey the orders given it.

自注意力：基本思路

- We multiply each value by its score and sum them up.

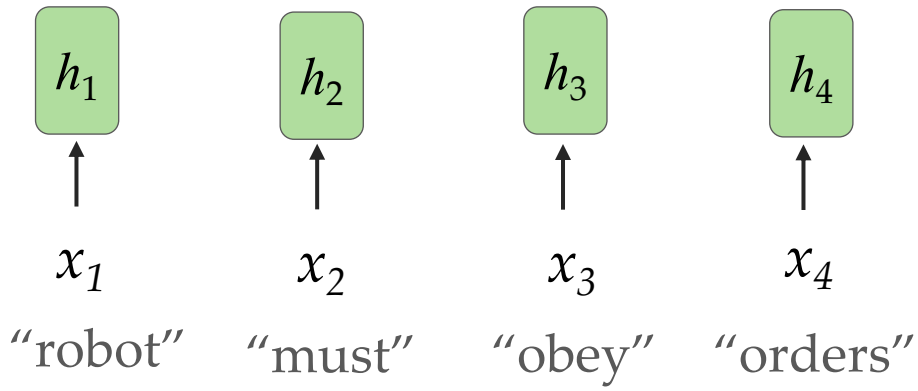
Word	Value vector	Score	Value X Score
<s>		0.001	
a		0.3	
robot		0.5	
must		0.002	
obey		0.001	
the		0.0003	
orders		0.005	
given		0.002	
it		0.19	
		Sum:	

- The outcome vector represents the new state (refreshed memory) for the query word.

步骤1: 词嵌入

- 将每个词元(表示为整数id) 转换为词向量 (看做初始隐含状态).

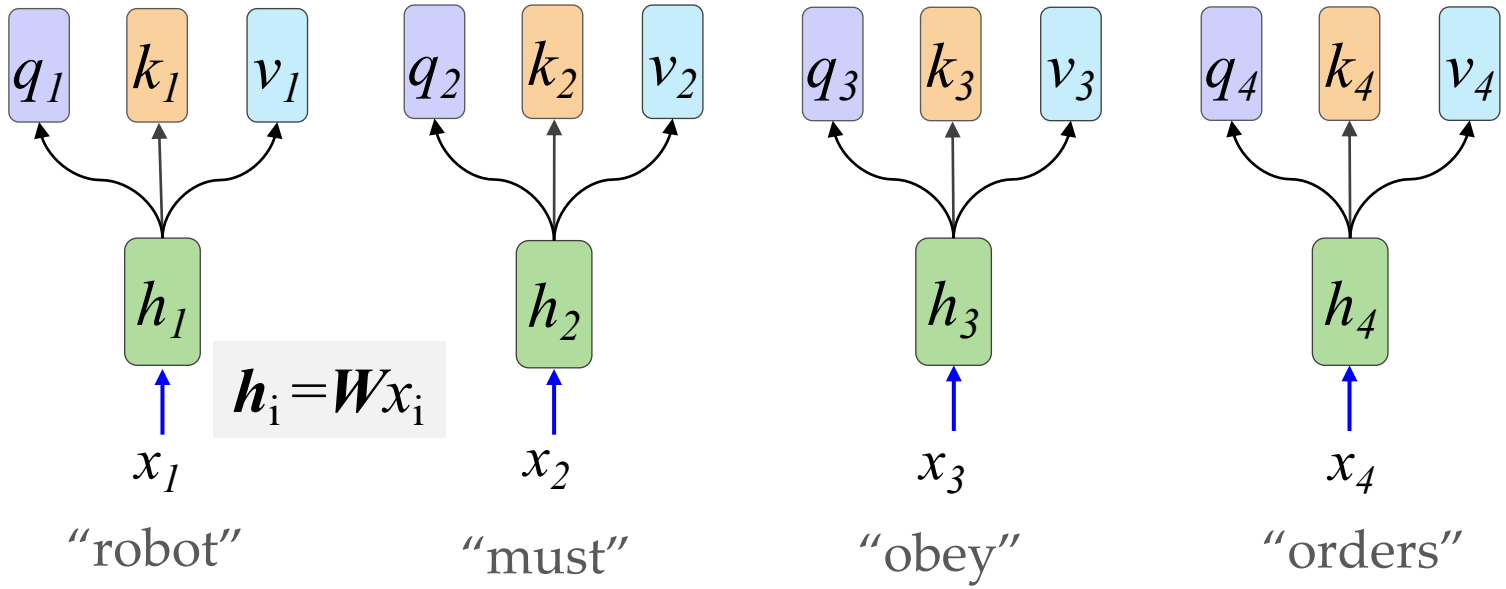
$$\boldsymbol{h}_i = \boldsymbol{W} x_i$$



步骤2：计算Q, K, V 向量

- 将每个隐含状态转成query/key/value向量:

$$q_i = W^q h_i \qquad k_i = W^k h_i \qquad v_i = W^v h_i$$

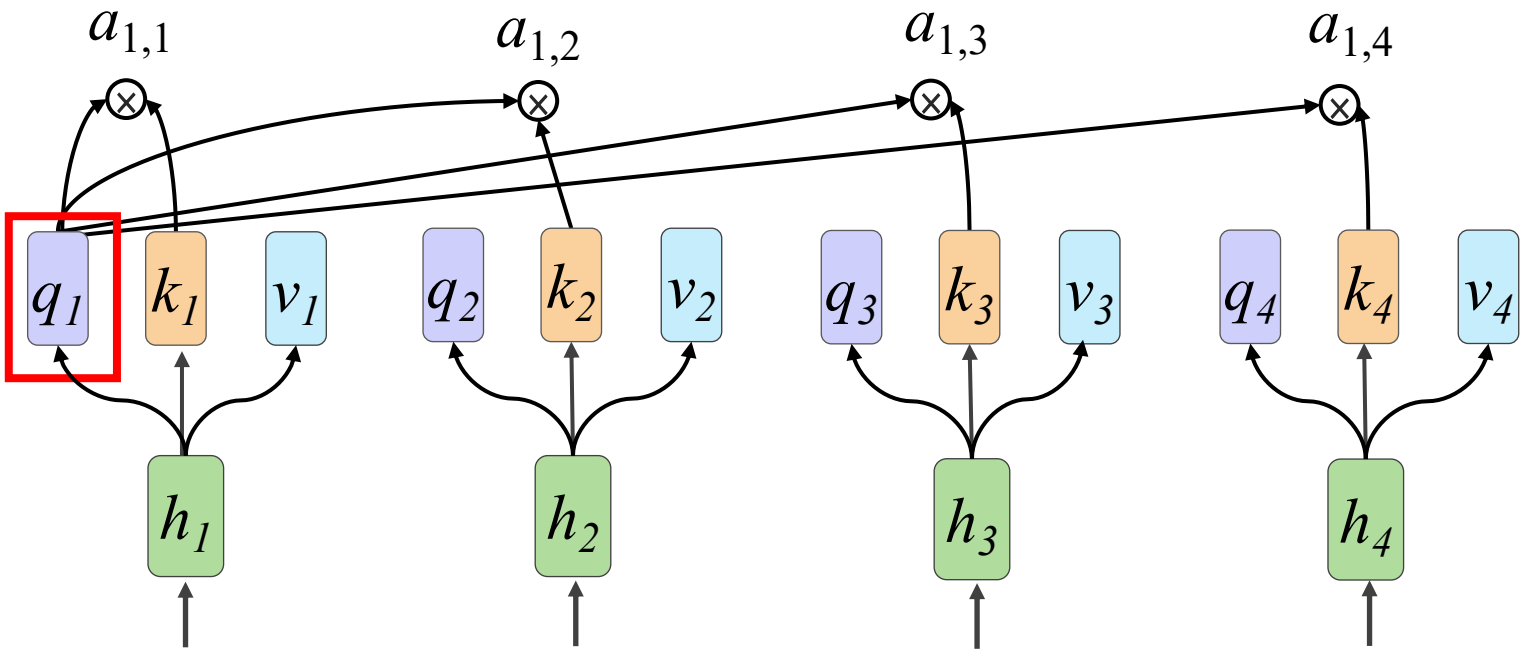


步骤3: 计算注意力分数

- 对每个<query, key> 对, 计算他们的**放缩内积**，作为它们的注意力分数。

$$a_{1,i} = \frac{q_1{}^T k_i}{\sqrt{d}}$$

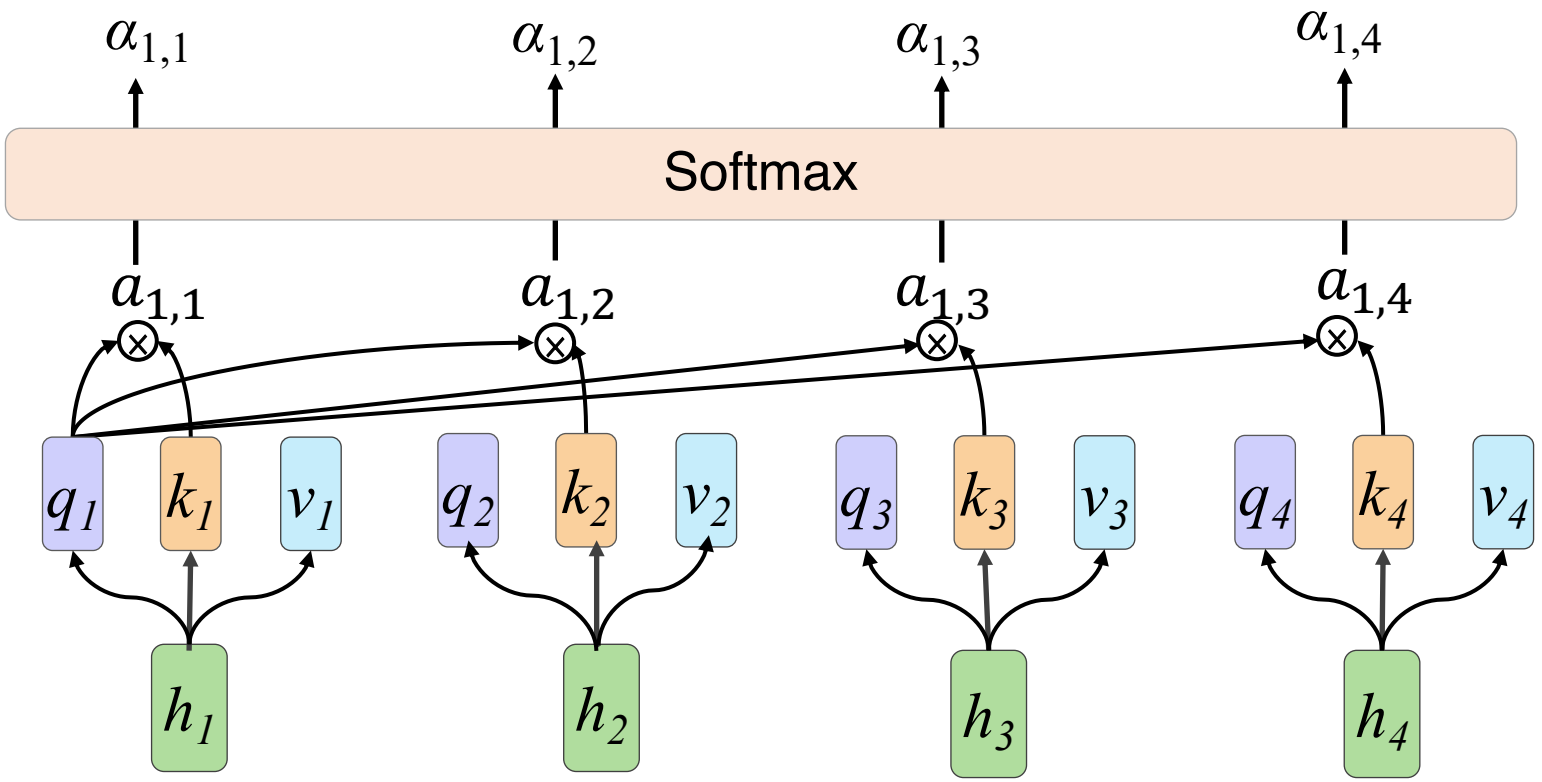
where d is the dim of q and k



步骤4：归一化分数

- 通过Softmax将注意力分数归一化成注意力权重(0~1之间的概率)

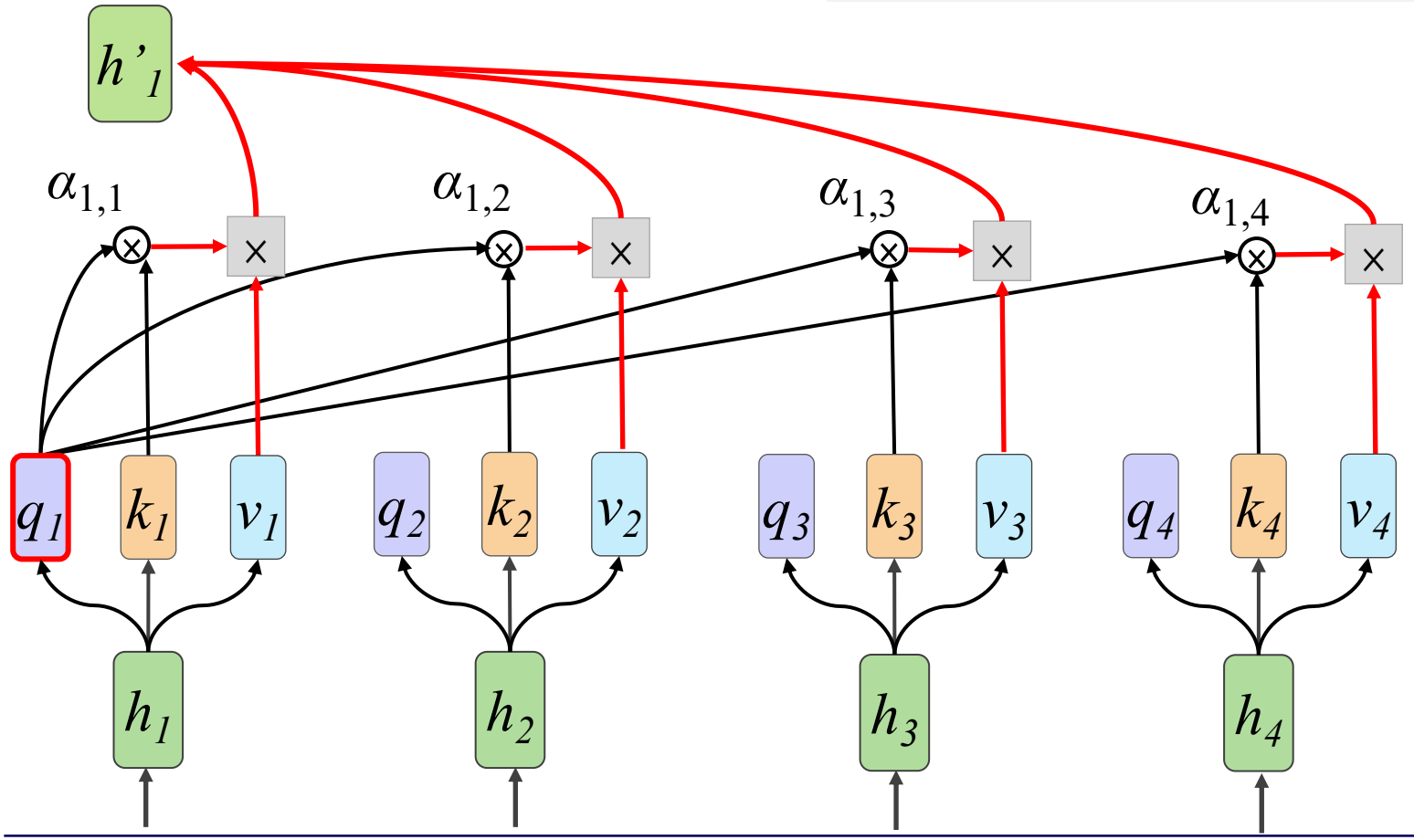
$$\alpha_{1,1},\ldots,\alpha_{1,T}=\text{softmax}(a_{1,1},\ldots,a_{1,T})$$



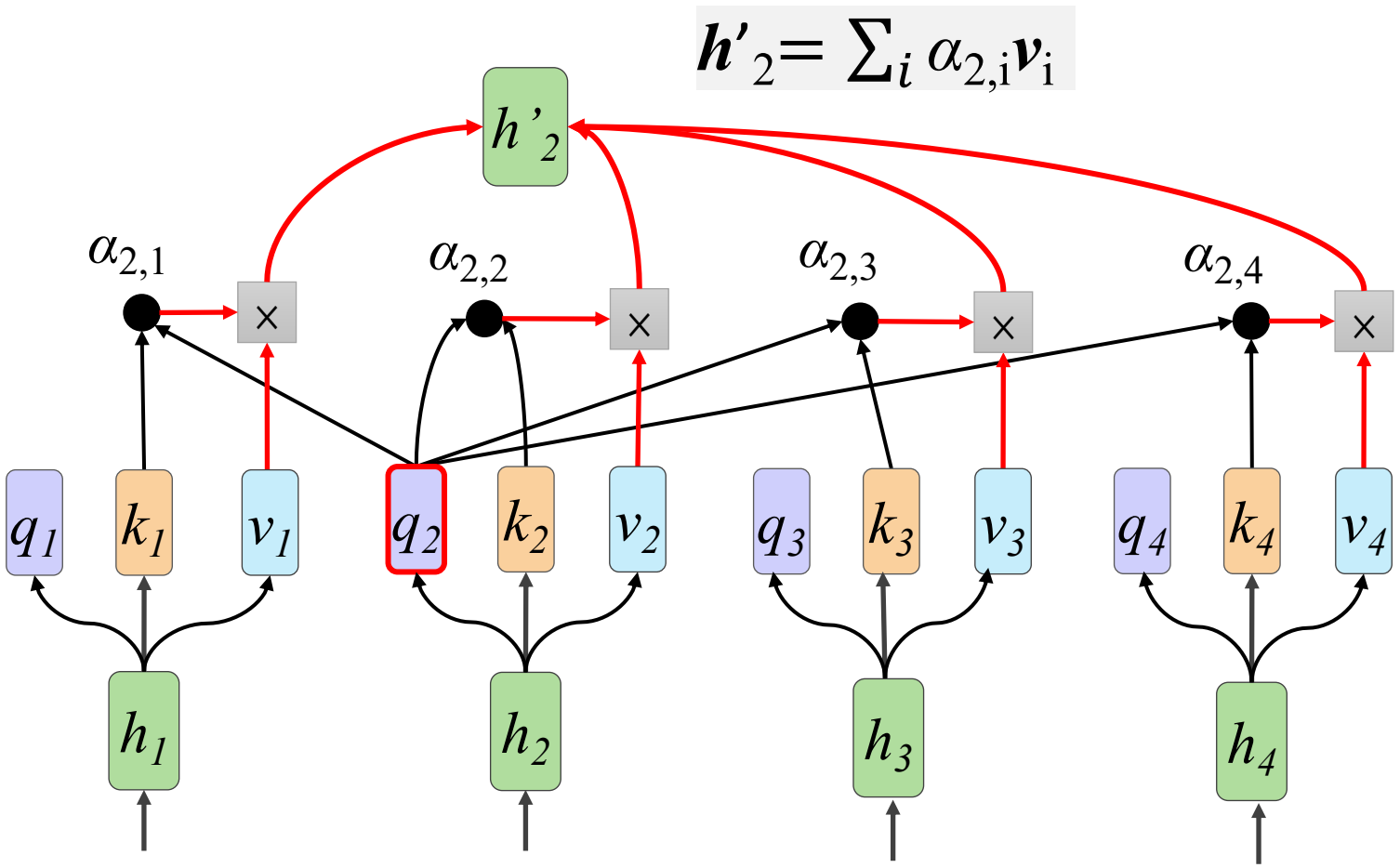
步骤5: 按照注意力权重将各个单词的状态(Value)聚合

$$h'_1 = \sum_i \alpha_{1,i} v_i$$

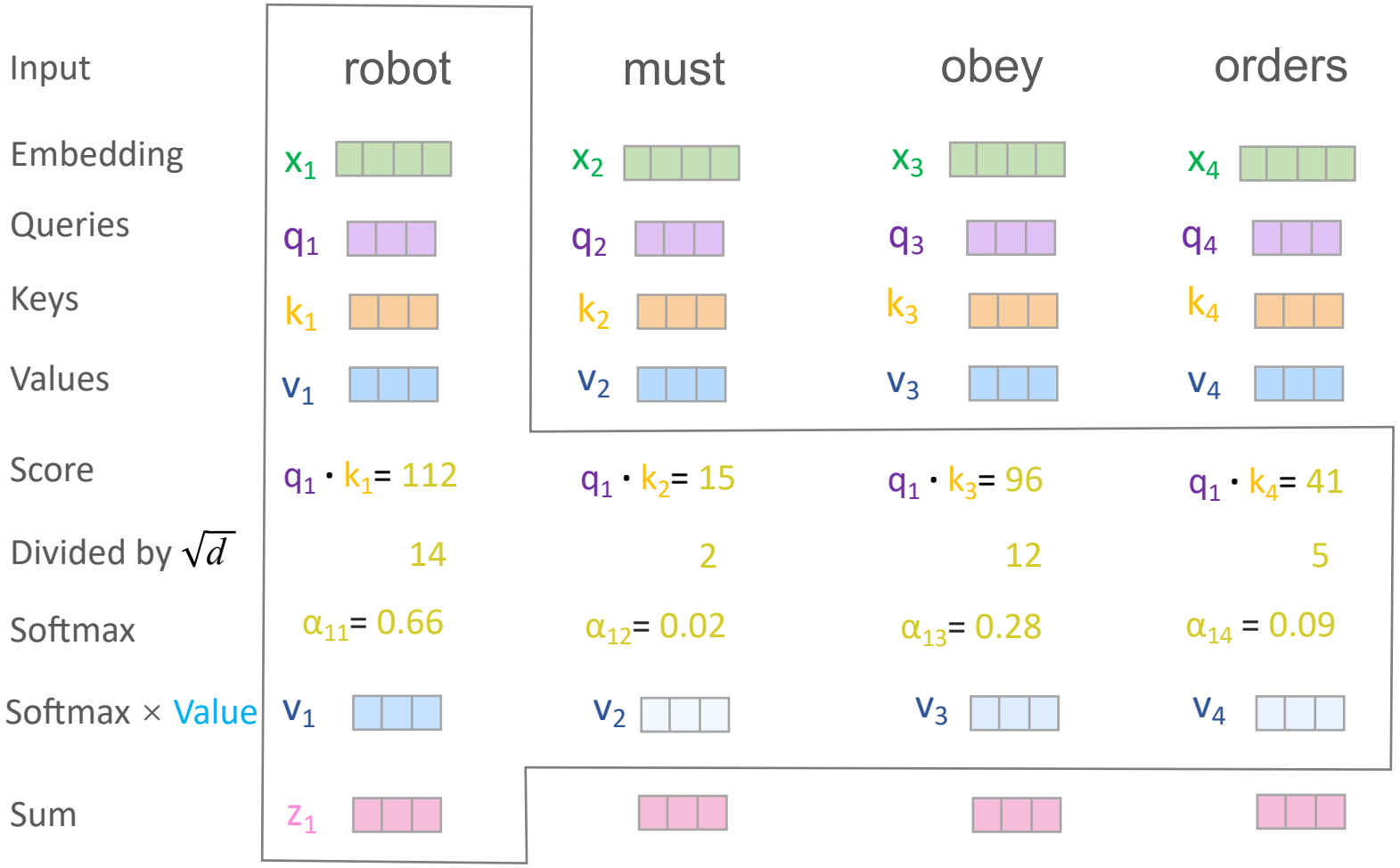
Consider the entire sequence…



步骤5: 按照注意力权重将各个单词的状态(Value)聚合



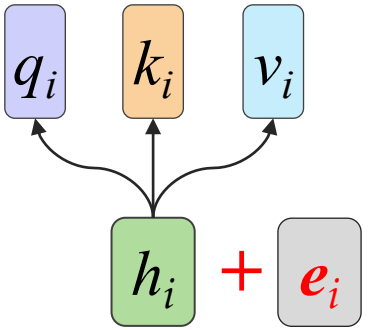
注意力计算举例

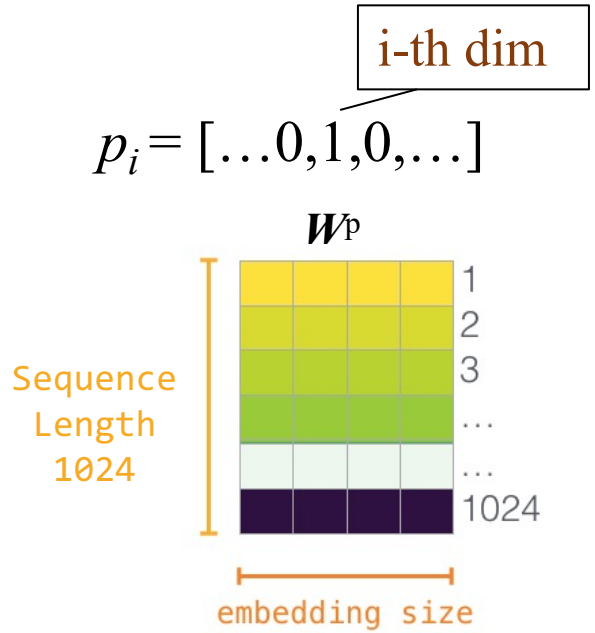

Input
robot
must
obey
orders
Embedding
x_1
x_2
x_3
x_4
Queries
q_1
q_2
q_3
q_4
Keys
k_1
k_2
k_3
k_4
Values
v_1
v_2
v_3
v_4
Score
$q_1 \cdot k_1 = 112$
$q_1 \cdot k_2 = 15$
$q_1 \cdot k_3 = 96$
$q_1 \cdot k_4 = 41$
Divided by $\sqrt{d}$
14
2
12
5
Softmax
$\alpha_{11} = 0.66$
$\alpha_{12} = 0.02$
$\alpha_{13} = 0.28$
$\alpha_{14} = 0.09$
Softmax × Value
v_1
v_2
v_3
v_4
Sum
z_1

如何表示语句顺序?

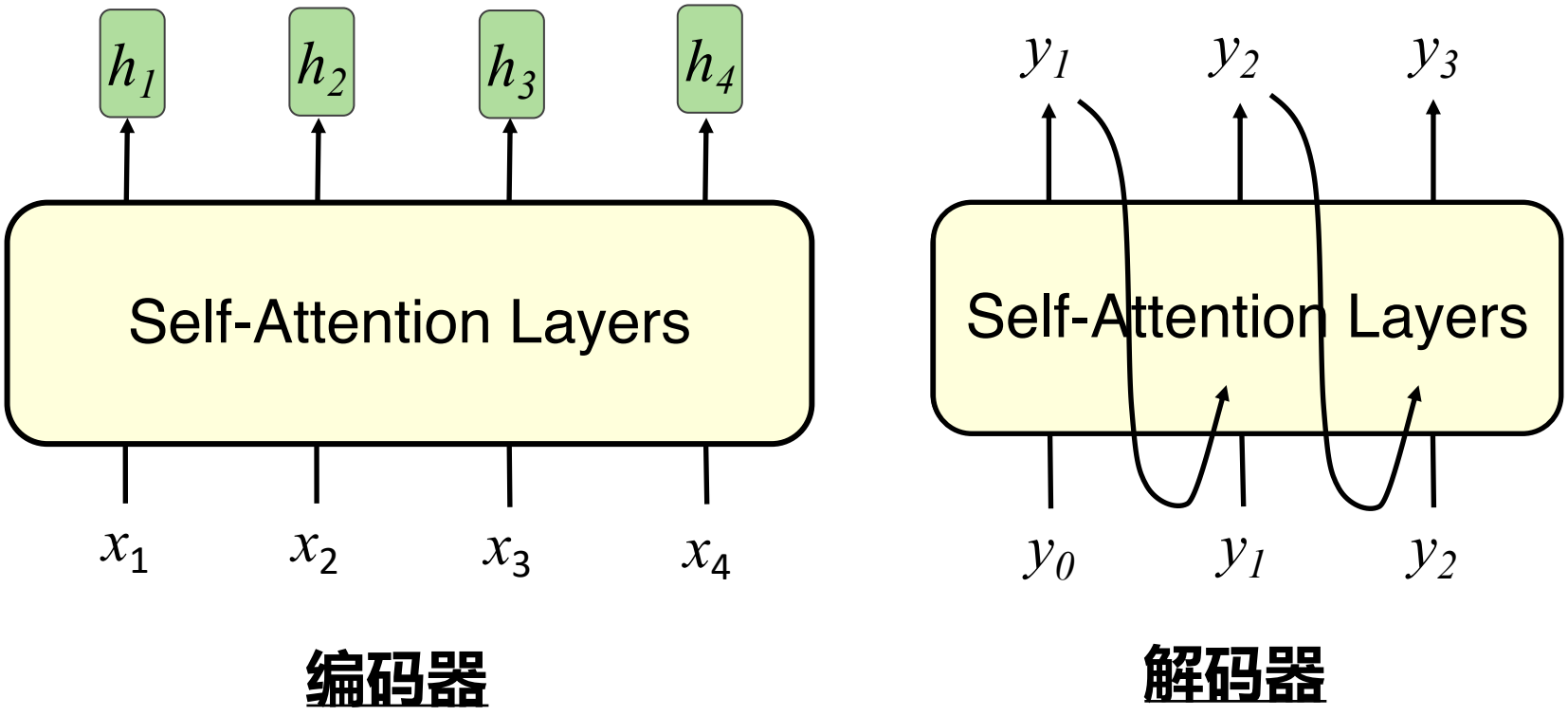
- **位置编码**: each position is assigned with a unique positional embedding $\boldsymbol{e}_i$
- $\boldsymbol{e}_i$ is added to the word embedding $\boldsymbol{h}_i$, as an input to the self-attention layer.
- let $\boldsymbol{p}_i$ be a **one-hot** vector indicating the position of x_i in the sequence, $\boldsymbol{e}_i$ can be obtained from a **learnable** embedding matrix $\boldsymbol{W}^p$

$$\boldsymbol{e}_i = \boldsymbol{W}^p \boldsymbol{p}_i$$





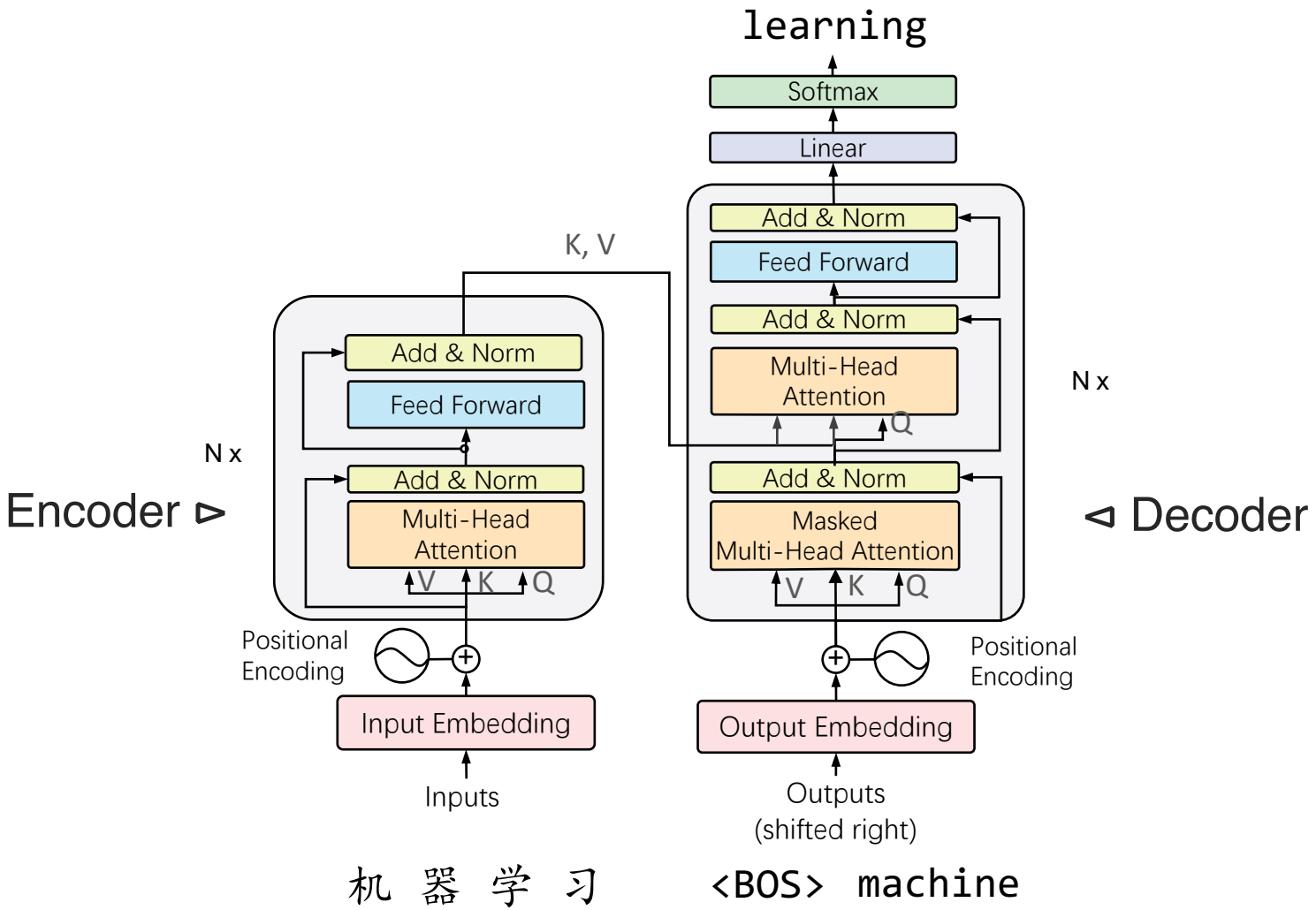
Seq2seq ~~with~~Using Self-Attention



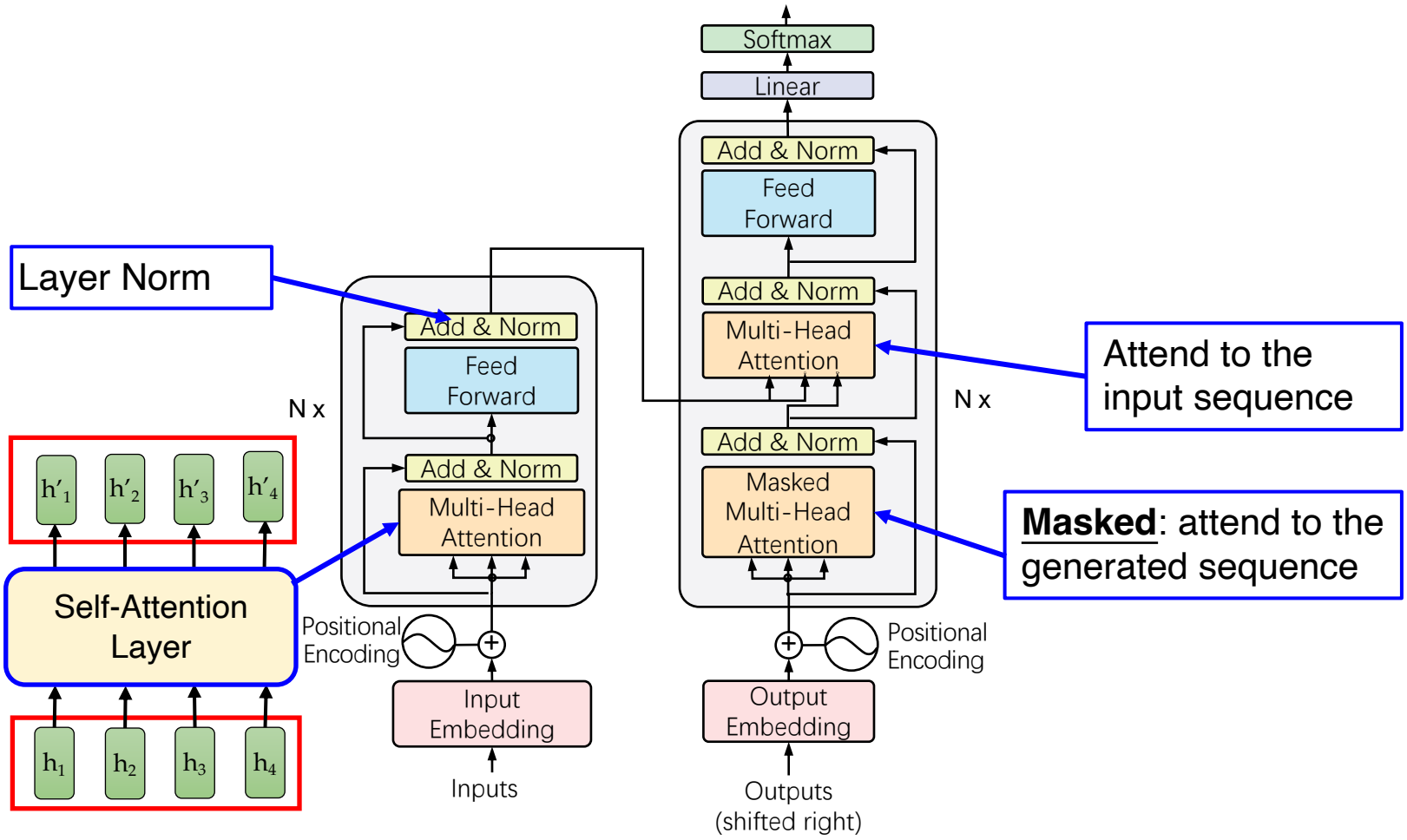
编码器 **解码器**

Ashish Vaswani, et al. “Attention Is All You Need”. ICLR 2017

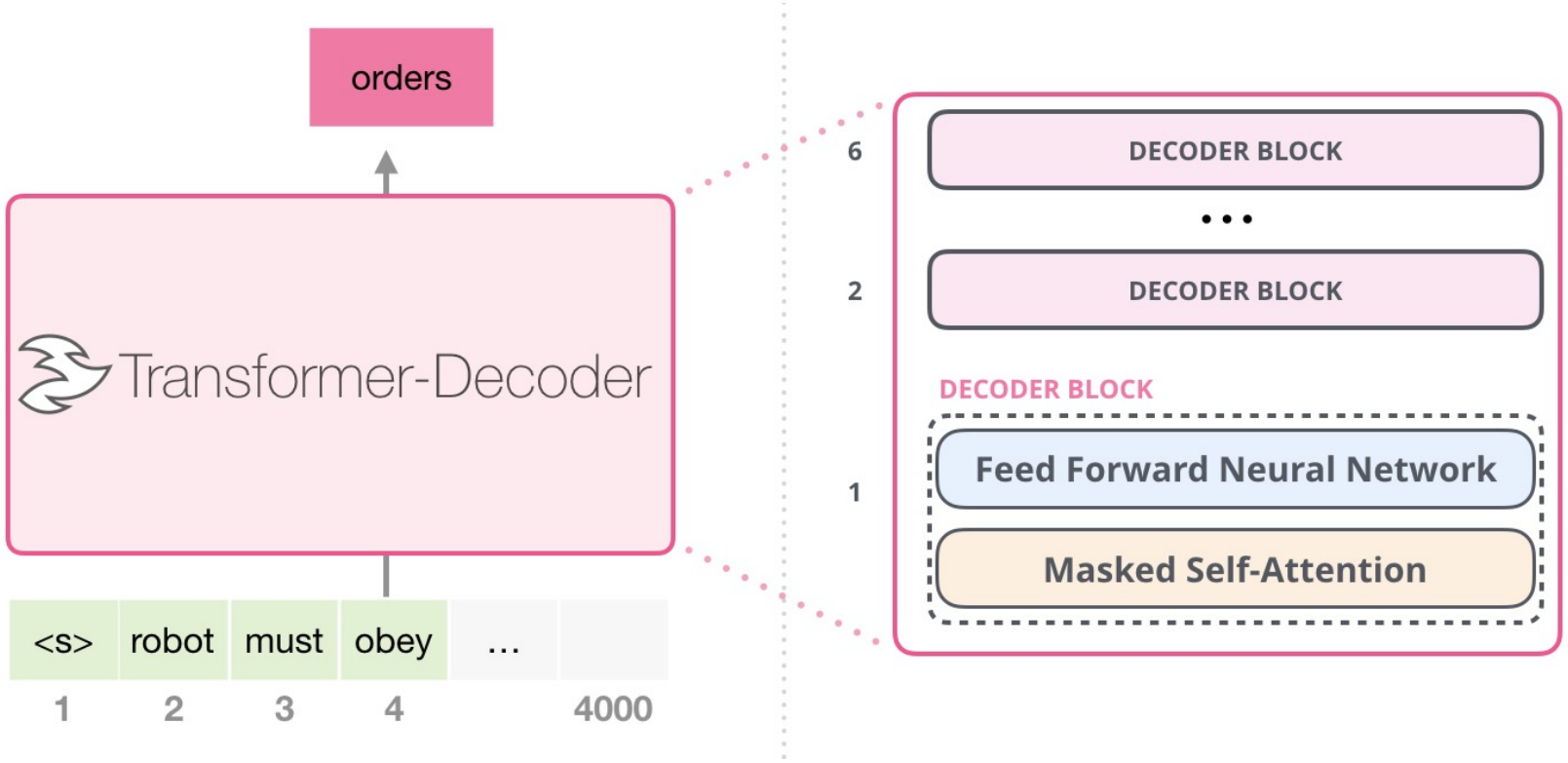
Transformer

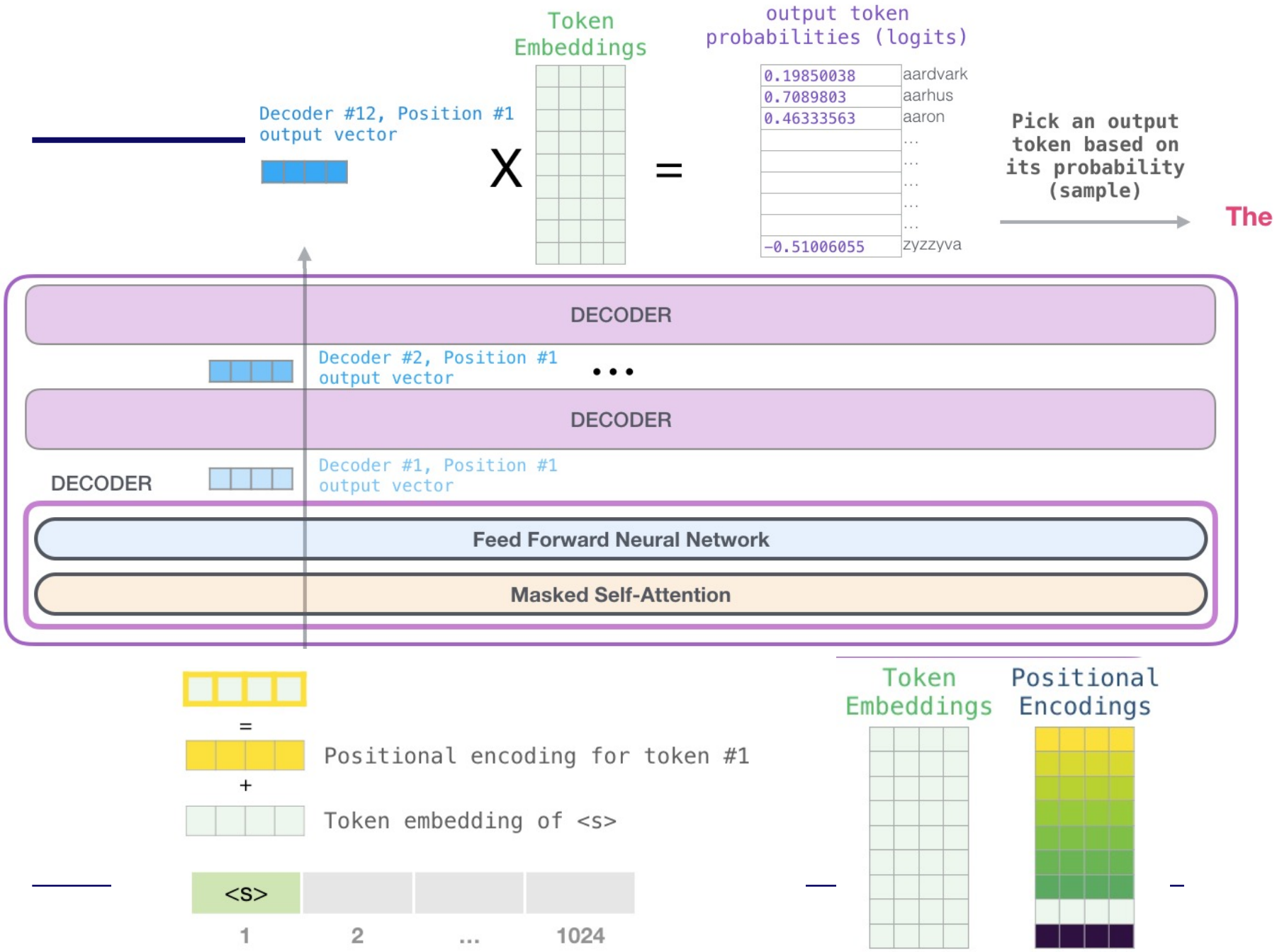


模型结构

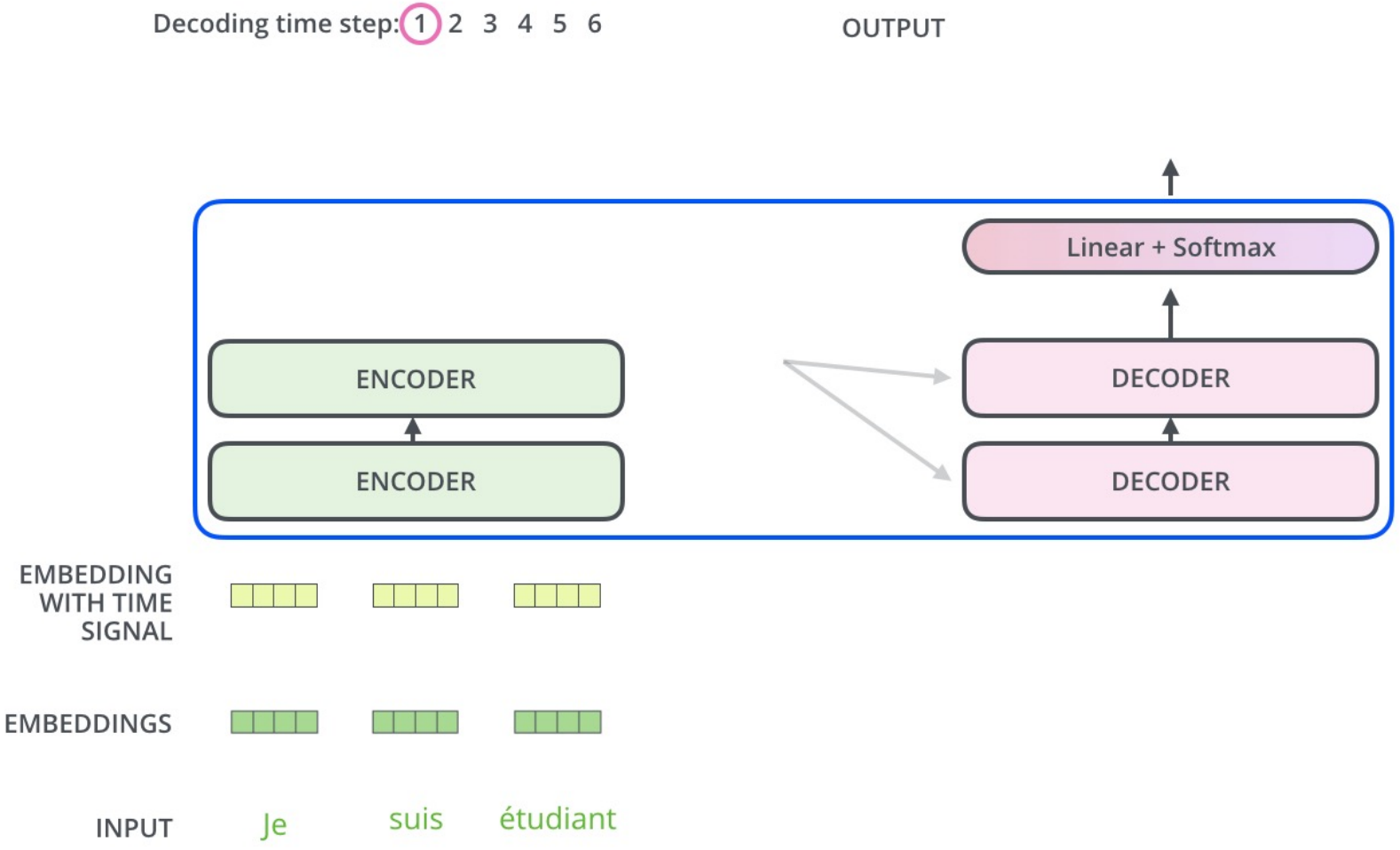

Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
N x
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
N x
Layer Norm
Attend to the input sequence
Masked: attend to the generated sequence
Self-Attention Layer
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

解码器详解

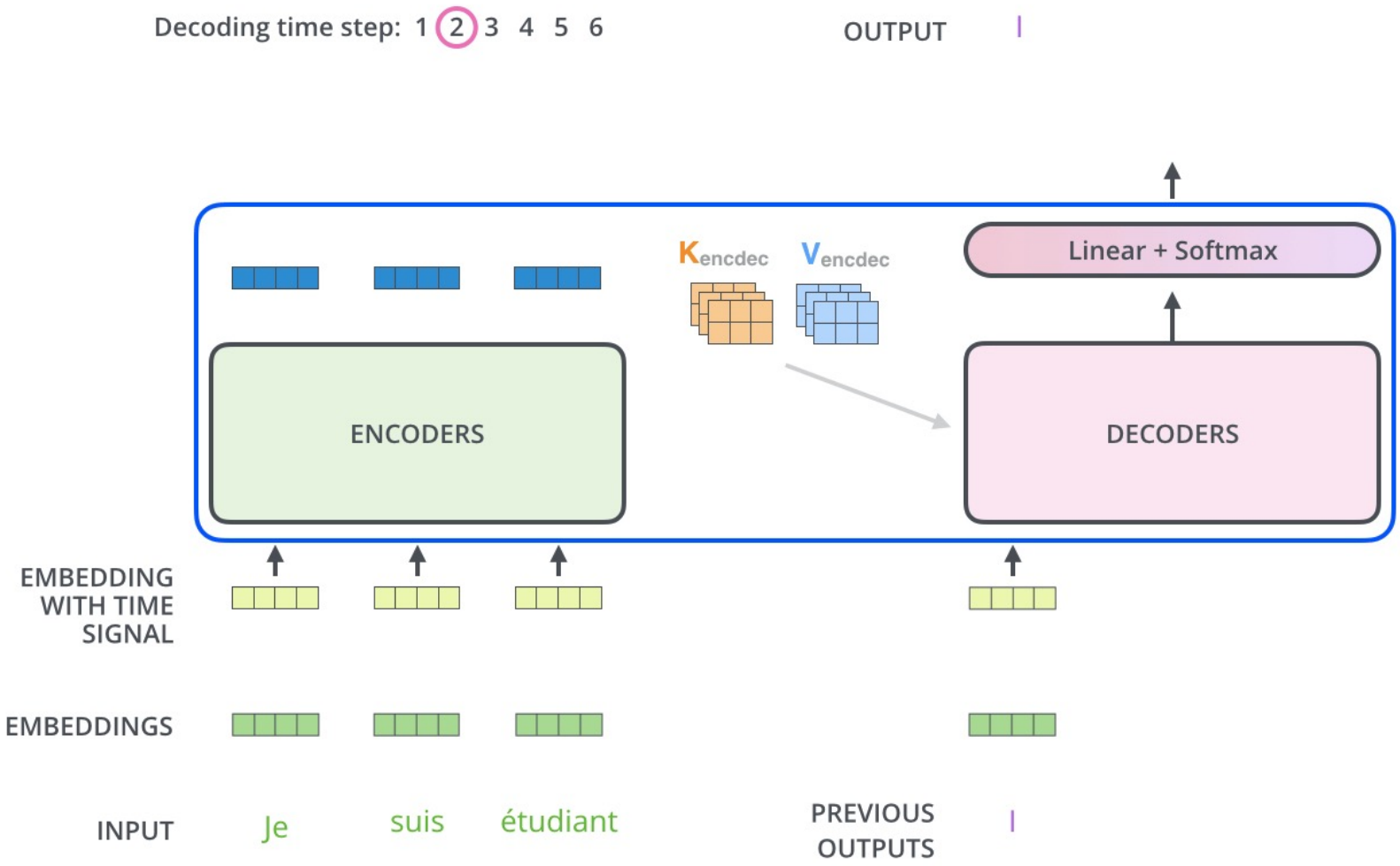

orders
Transformer-Decoder
<s>
robot
must
obey
…
1
2
3
4
4000
6
2
1
DECODER BLOCK
DECODER BLOCK
DECODER BLOCK
Feed Forward Neural Network
Masked Self-Attention


Token Embeddings
output token probabilities (logits)
Decoder #12, Position #1 output vector
X
=
0.19850038 aardvark
0.7089803 aarhus
0.46333563 aaron
-0.51006055 zyzzyva
Pick an output token based on its probability (sample)
The
DECODER
Decoder #2, Position #1 output vector
DECODER
Decoder #1, Position #1 output vector
DECODER
Feed Forward Neural Network
Masked Self-Attention
=
Positional encoding for token #1
+
Token embedding of <s>
<s>
1
2
...
1024
Token Embeddings
Positional Encodings

解码器详解

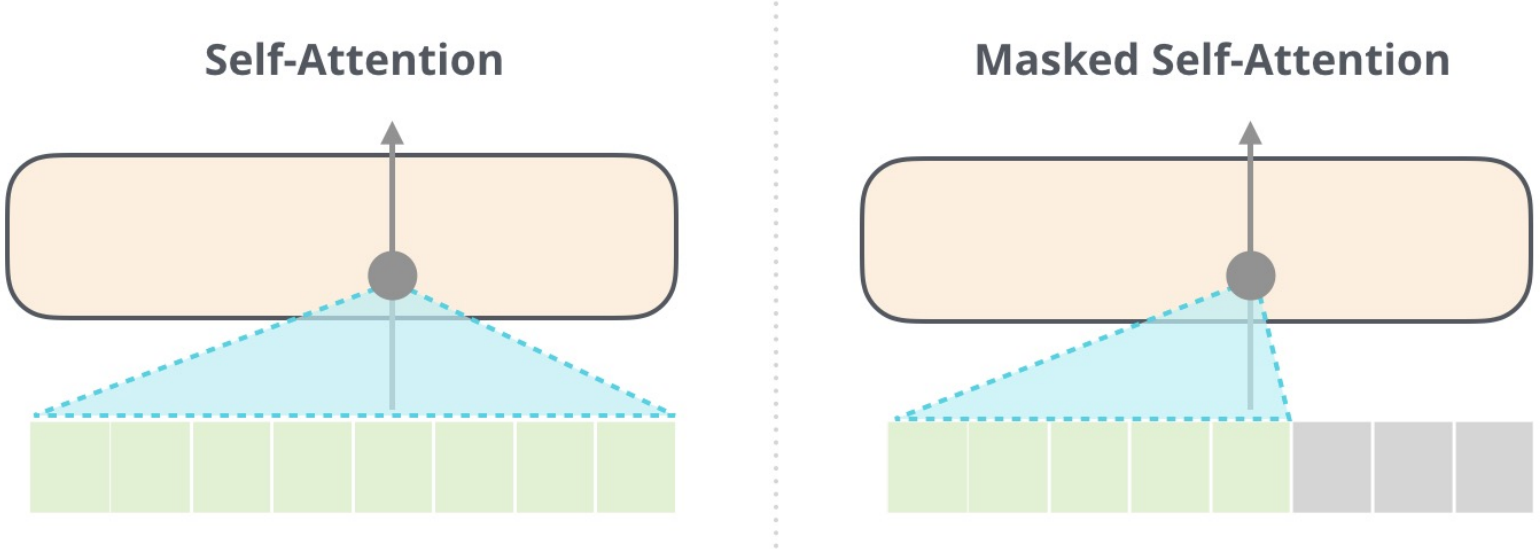

Decoding time step: 1 2 3 4 5 6
OUTPUT
Linear + Softmax
ENCODER
ENCODER
DECODER
DECODER
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant

解码器详解


Decoding time step: 1 2 3 4 5 6
OUTPUT
I
Linear + Softmax
K_{encdec}
V_{encdec}
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant
PREVIOUS OUTPUTS
I

注意力掩码

- 一个大小为L×L 的矩阵，作用在注意力分数上
- 避免模型注意到指定位置的单词



注意力掩码


Attention Scores
robots must obey rules
Self-Attention
robots 0.11 0.00 0.81 0.79
must 0.19 0.50 0.30 0.48
obey 0.53 0.98 0.95 0.14
rules 0.81 0.86 0.38 0.90
Attention Mask
robots must obey rules
robots 0 -inf -inf -inf
must 0 0 -inf -inf
obey 0 0 0 -inf
rules 0 0 0 0
=
Masked Scores
robots must obey rules
robots 0.11 -inf -inf -inf
must 0.19 0.50 -inf -inf
obey 0.53 0.98 0.95 -inf
rules 0.81 0.86 0.38 0.90
Softmax (along rows)
Attention Weights
robots must obey rules
robots 1 0 0 0
must 0.48 0.52 0 0
obey 0.31 0.35 0.34 0
rules 0.25 0.26 0.23 0.26

注意力掩码

Self-Attention Mask

- for both encoder and decoder
- to ignore padding tokens

Cross-Attention Mask

- for decoder only
- to ignore padding tokens in the source sequence

Causal Mask

- for decoder only
- only pay attention to generated words

	robots	must	obey	orders	<pad>
robots	0	0	0	0	-inf
must	0	0	0	0	-inf
obey	0	0	0	0	-inf
orders	0	0	0	0	-inf
<pad>	0	0	0	0	-inf

	robots	must	obey	orders
robots	0	-inf	-inf	-inf
must	0	0	-inf	-inf
obey	0	0	0	-inf
rules	0	0	0	0

Transformer动态演示

https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

训练方法

和RNN编解码模型一样

数据: $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})\}$,

其中 $x^{(\ell)} = (x_1, \ldots x_{Tx}), y^{(\ell)} = (y_1, \ldots y_{Ty})$.

损失函数 – 最小化交叉熵损失函数

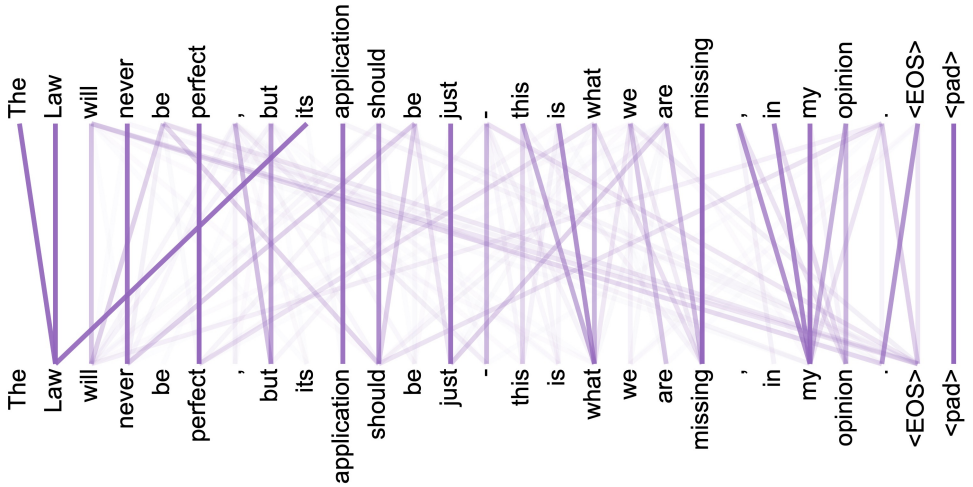
$$l(\theta|x, y) = -\sum_{t=1}^{T} \log p_\theta(y_t|y_1, \ldots, y_{t-1}, x)$$

$$L(\theta|D) = -\frac{1}{N}\sum_{l=1}^{N} l(\theta|x^{(l)}, y^{(l)})$$

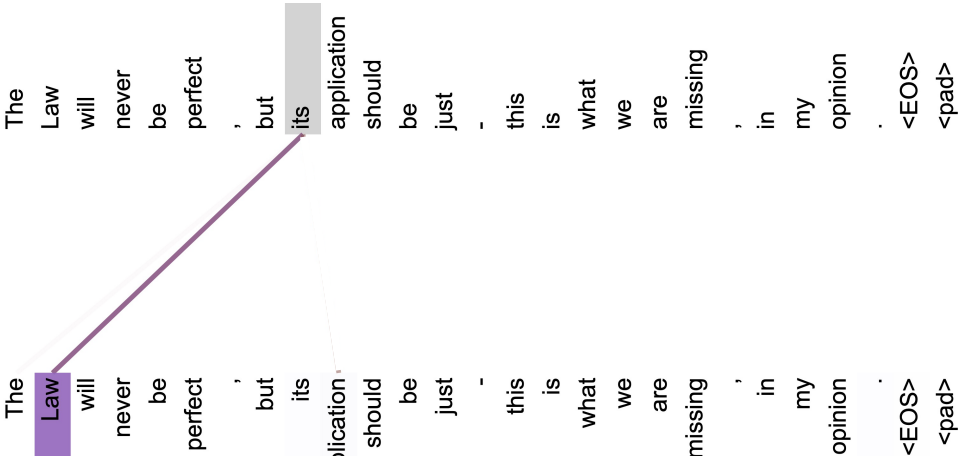
优化算法 – 梯度下降

注意力可视化

Full attentions in layer 5 of 6 for head 5.



Isolated attentions from just the word 'its' for attention heads 5. Note that the attentions are very sharp for this word.



https://arxiv.org/abs/1706.03762