



# 9. Transformer

孙晓光

[xgsun@fudan.edu.cn](mailto:xgsun@fudan.edu.cn)

2025-4-17



# 提 纲

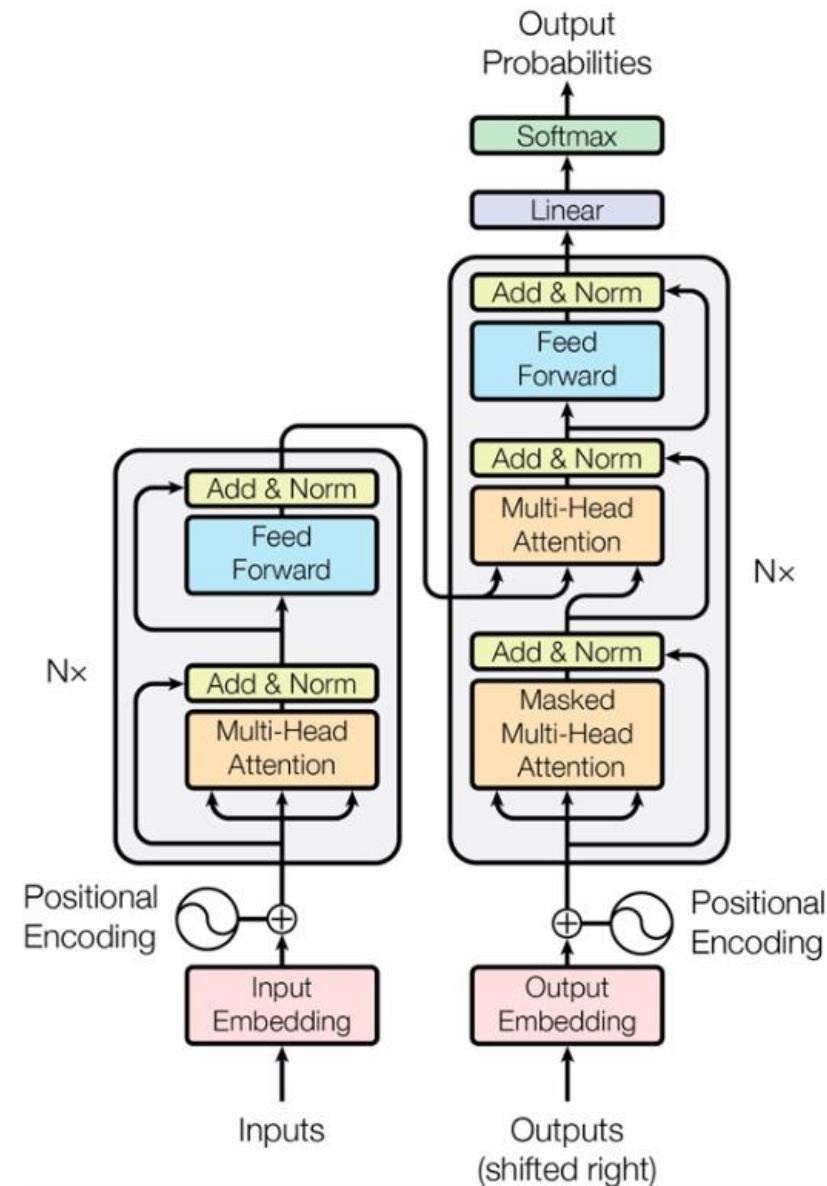
① Seq2Seq

② 注意力 机制

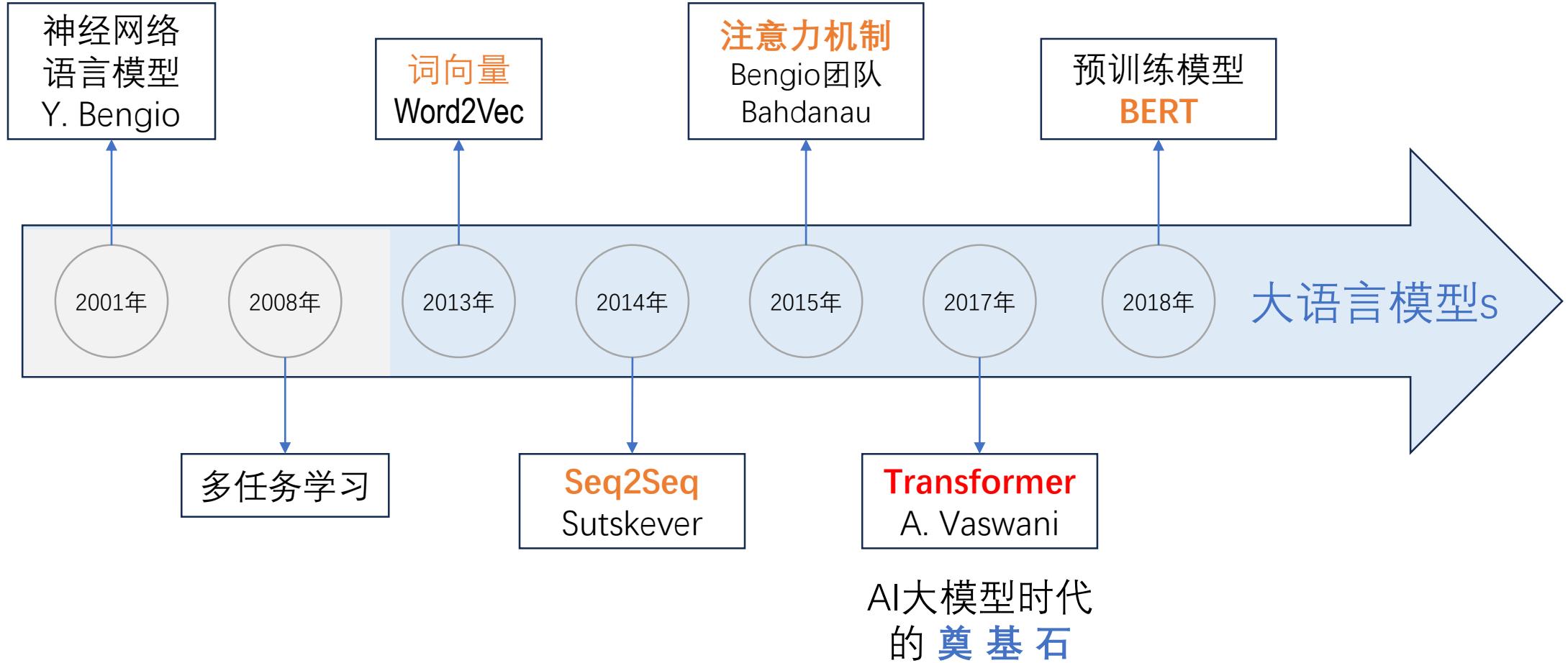
③ Transformer 架构

④ BERT

⑤ GPT



# 现代 NLP 里程碑

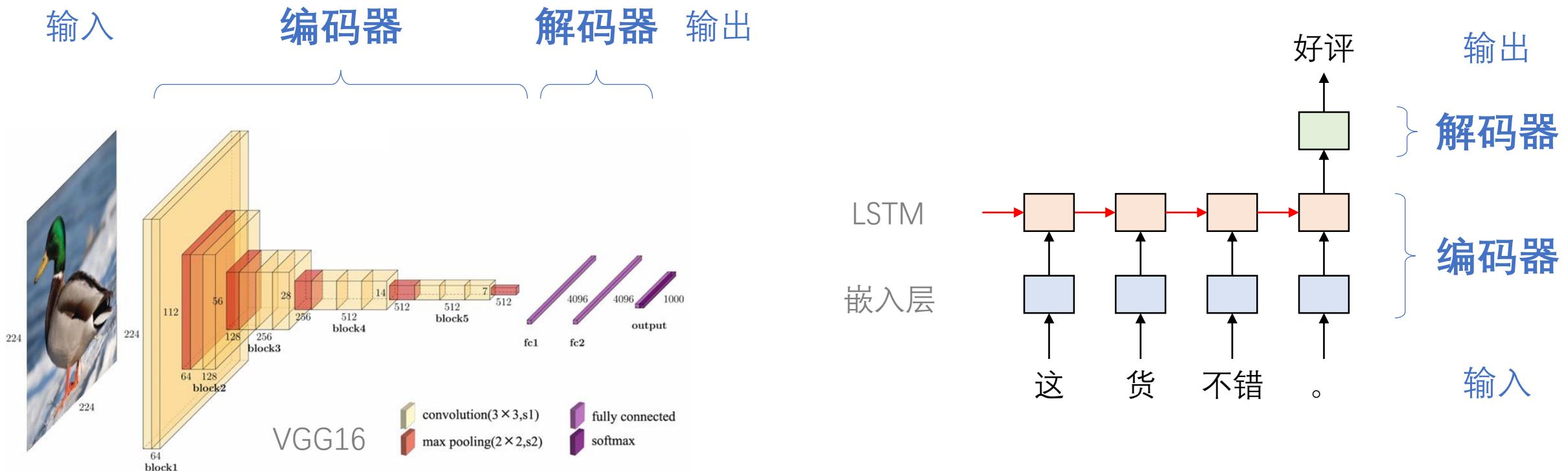


# 1

# Seq2Seq

# CNN

# RNN



**编码器：**将输入(图像)表示成中间表达式(特征)

**解码器：**将中间表达式表示(解码)成输出

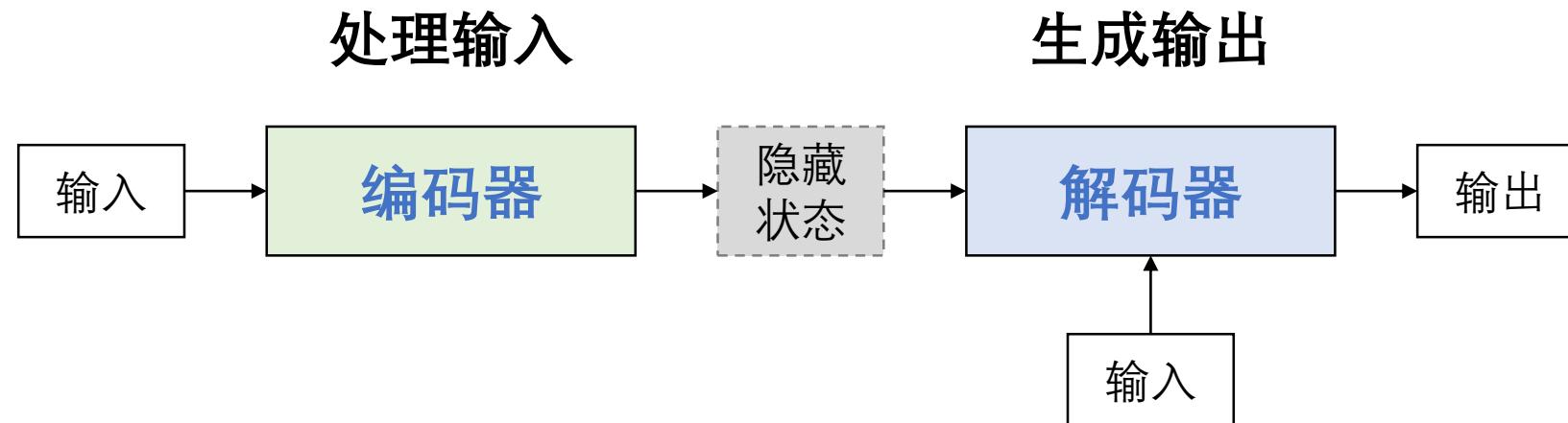
**编码器：**将输入(文本)表示成向量

**解码器：**将向量表示成输出

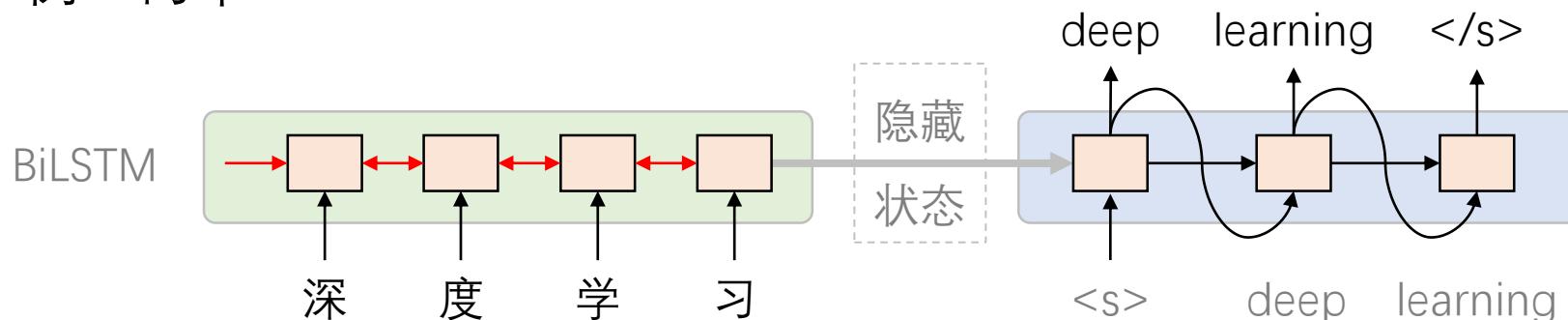
encoder

# 编码器 - 解码器

decoder



机器翻译示例：两个RNN

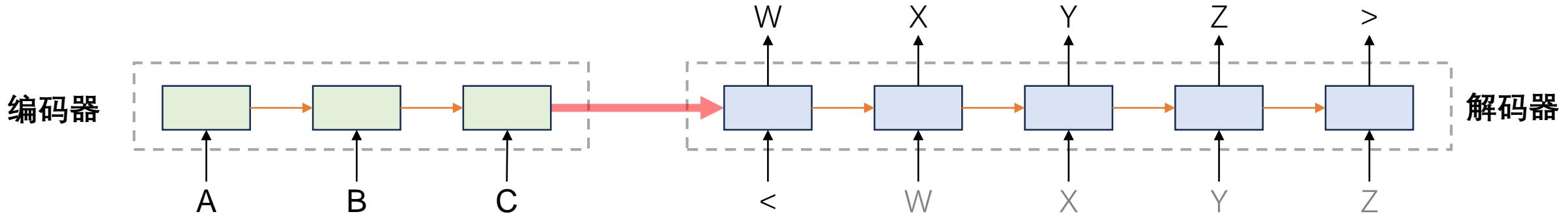


将整个序列的信息  
压缩成一个向量表示

上下文  
语义向量  
容量小!

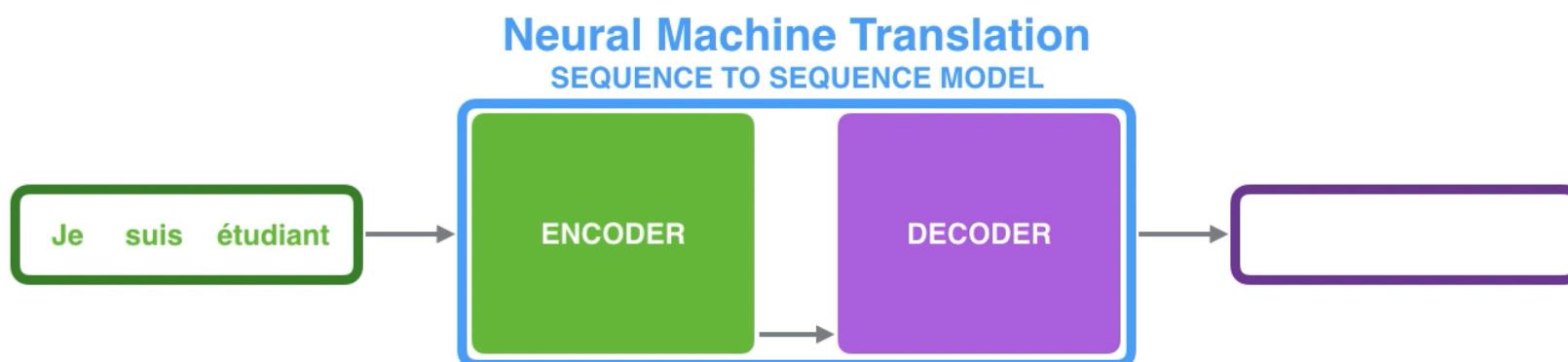
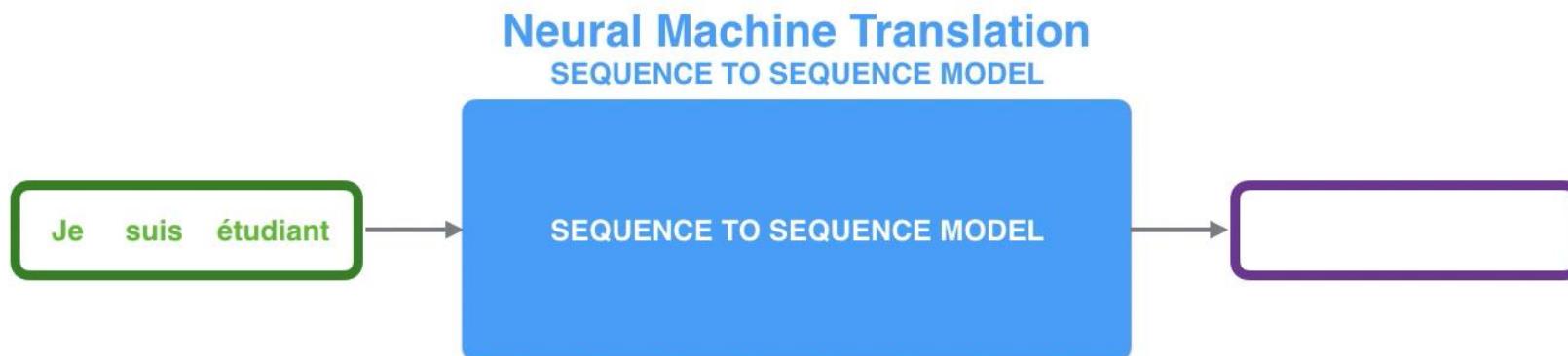
解决了输入序列长度与  
输出序列长度不同的问题

# Seq2Seq 架构



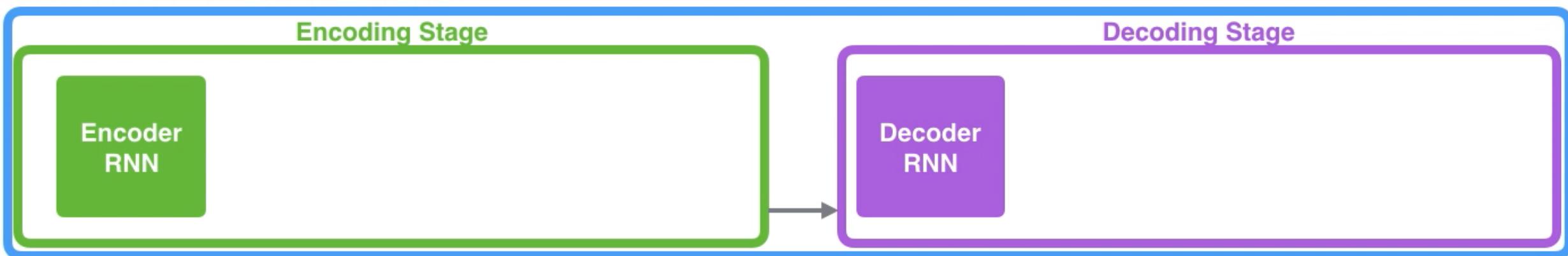
- 输入一个序列，生成另一个序列。
- 可由两个独立的RNN组成：
  - 编码器：输入多个时间步长信息，编码为**上下文语义向量**；
  - 解码器：将稳定状态解码为输出序列。**反向读取输入句子，短期依赖。**
- **突破**：传统的**固定长短输入问题**。
- **局限**：**上下文语义向量**对于解码阶段每个时间步长都是一样的。
- 应用：聊天机器人、语音识别、对话系统、问答系统.....

# Seq2Seq



# 编码器、解码器

## Neural Machine Translation SEQUENCE TO SEQUENCE MODEL



Je suis étudiant

# 2

# 注意力 机制

# 人的注意力

- 人脑每个时刻接收的外界输入信息非常多：视觉、听觉、触觉。
- 仅视觉：眼睛每秒钟发送千万比特的信息给视觉神经系统。
- 人脑通过**注意力**来解决**信息超载问题**。

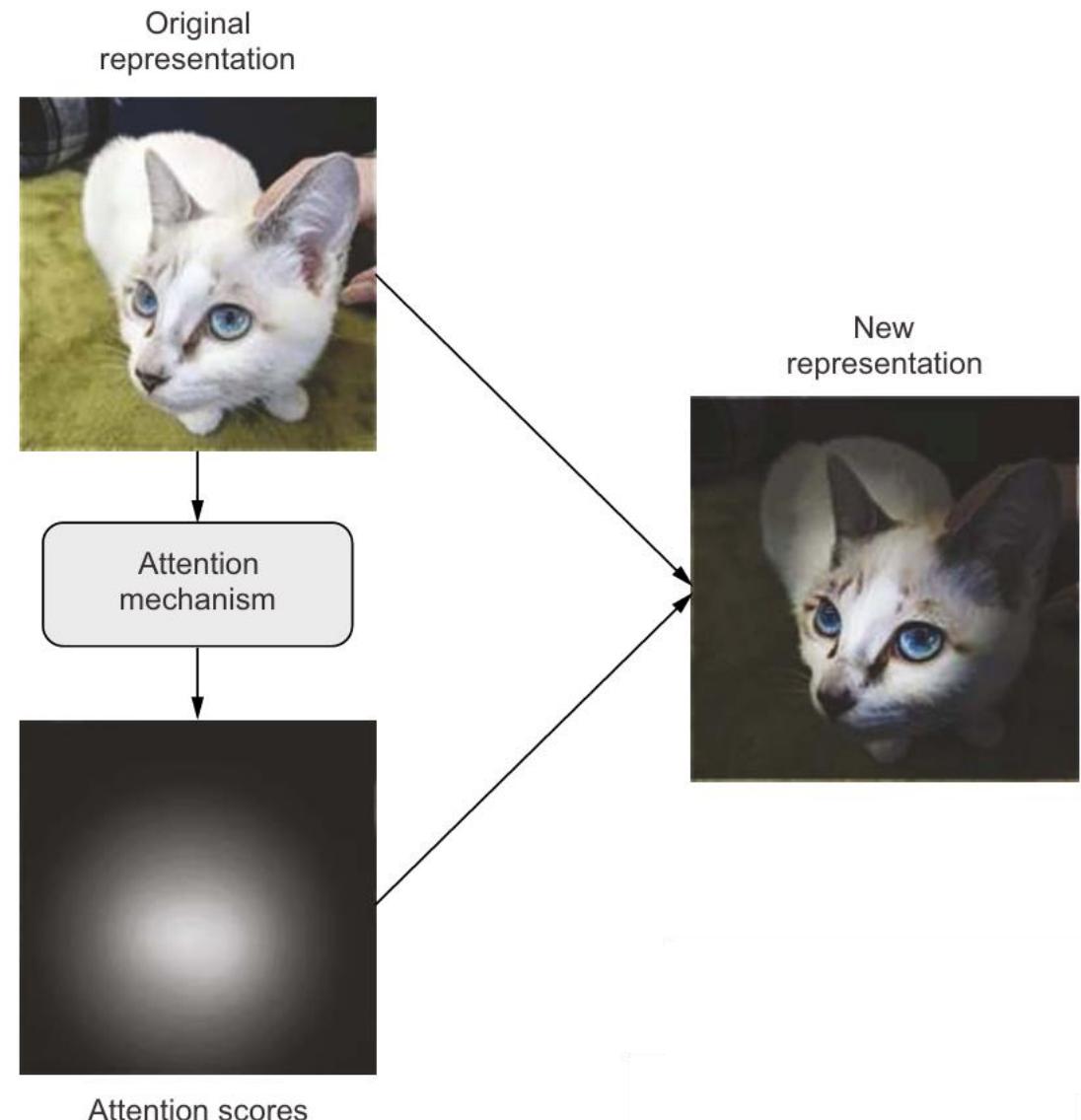
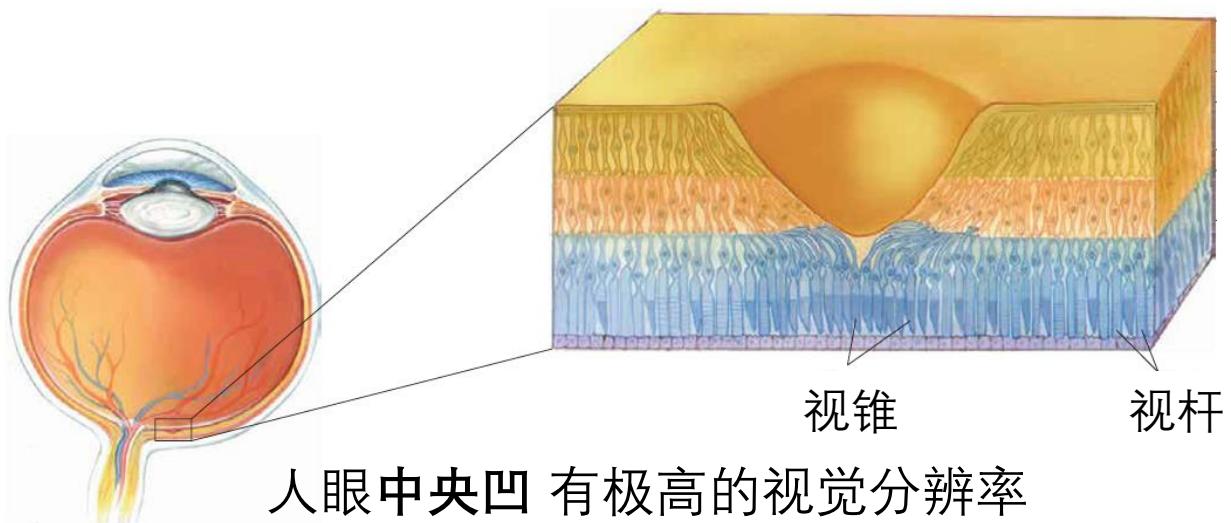


高度关注  
我正在 吃 绿色的 苹果。  
低度关注

# 注意力 Attention

## 多种注意力方法

- CNN中的最大池化：保留一个最重要特征，舍弃其他特征。全有/全无
- SENet：注意力集中在重要的特征图通道上。



# 两种 注意力

认知心理学：专注于研究人类如何获取、加工、存储和使用信息，  
核心问题：记忆、**注意力**、语言、问题解决 .....

- **注意力**：认知系统对有限资源的分配机制，从海量信息中选择关键内容。
- 从驱动来源：**外部刺激驱动的无意识注意力**、**内部目标驱动的有意识注意力**。



视觉“弹出效应”



鸡尾酒会效应

# 注意力机制

Attention Mechanism

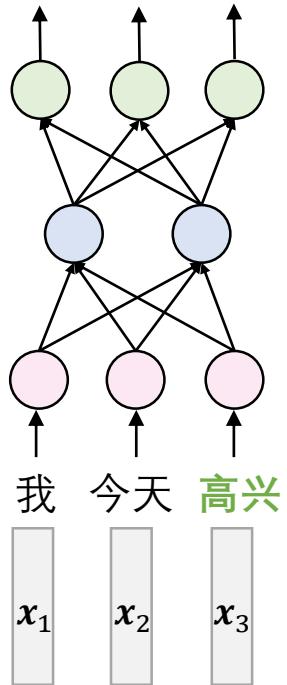
- 模仿人类认知注意力 (动态分配计算资源, 聚焦关键信息) 的计算模型。
- 广泛应用: 自然语言处理、计算机视觉、多模态任务 .....
- 核心思想:** 通过输入数据的**权重分布**, 决定哪部分需被重点关注。
- 数学本质:** 对输入信息的**加权求和**。

$$\text{输出} = \sum_i \alpha_i \cdot \text{输入}_i$$

权重  $\alpha_i$ : 由当前任务状态、输入信息的性质来匹配  
输入  $i$  : 对应的输入内容

# 注意力 网络

无 注意力机制：



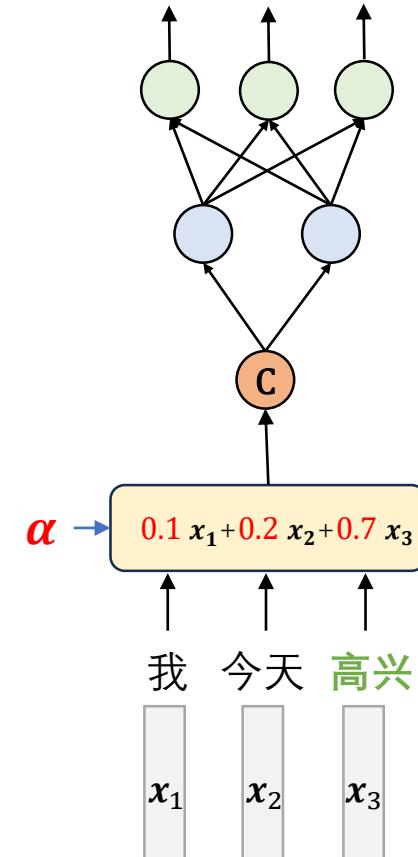
所有输入向量被平等对待，  
无法动态聚焦关键信息，  
如“高兴”可能对当前任务更重要。

$$\alpha = [0.1 \ 0.2 \ 0.7]$$

如何添加 注意力？

给“高兴”更大的权重  $\alpha$ ！

包含上下文信息的向量：  $c = \sum_i \alpha_i \cdot x_i$



# 注意力 权重 $\alpha$

$$\alpha_i = \text{Softmax}(W_{att} \cdot x_i)$$

- 全连接层
- 直接通过  $x_i$  计算权重  $\alpha_i$
- 用向量**点积**衡量二者的匹配度 (**相似度**)

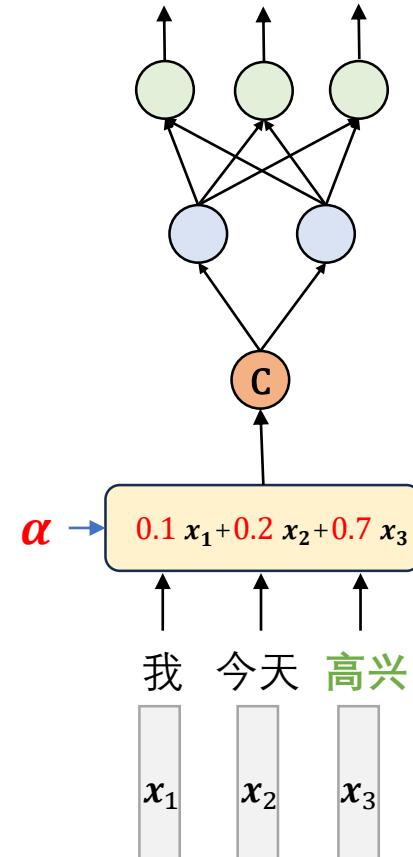
$$\mathbf{A} \cdot \mathbf{B} = \|\mathbf{A}\| \|\mathbf{B}\| \cos(\theta)$$

方向相似性：方向越接近，点积越大；方向相反点积越小。

方向相似性可反映语义或特征的关联性。

- $W_{att}$ ：可训练的注意力权值向量
- Softmax( )：归一化、概率分布

**问题：**  $x_i$  既表达**输入内容**，又表达**是否重要**。



# Q、K、V

- **Q** (你的需求) : “深度学习相关的书”
- **K** (书籍索引) : 每本书的标签, 如 “AI”、“数学”、“编程”...
- **V** (书籍内容) : 书的具体内容

你通过比较**需求(Q)**与**索引(K)**找到匹配的书 (**K**=“深度学习”),  
最终阅读的是这本书的内容 (**V**)。



代词	动作	对象
我	吃	苹果

→ | eat apples

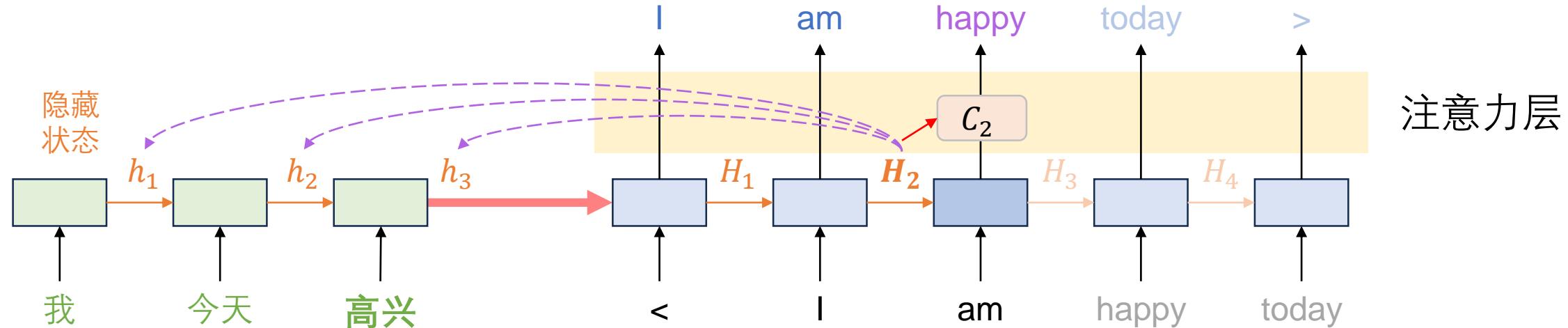
- **Q** (查询): 当前需要生成的目标词 (“eat”), 对源句子的需求“需要找一个表示**进食动作**的词”。像一个“探测器”, 主动提出需求。
- **K** (键) : 源句子每个词的“身份标签”, 等待被 **Q** 检索, 决定哪些位置与 **Q** 相关。通过计算: **Q** (“eat”) 和每个 **K** 的相似度, 发现 **Q** 与 **K** =“吃”的匹配度最高。
- **V** (值) : 来自源句子中每个词的内容。

# Q、K、V

- **Query**: 主动, 代表“当前需要关注什么”,  $Q = W_Q \cdot x_{query}$   
    **查询**   如, 取  $x_{query} = x_3$  作为当前焦点
  - **Key** : 被动, 与 Query 的相关性,  $K_i = W_K \cdot x_i$   
    **键**       表示“哪些位置重要? ”
  - **Value** : 内容, 需要提取的信息。  $V_i = W_V \cdot x_i$   
    **值**
- **Query** 代表任务目标, 动态指导注意力的方向, 通常来源于外部信息。
  - 将输入向量  $x_i$  分别映射为 **Key**、**Value**

$$[\alpha_0, \alpha_2, \dots, \alpha_n] = \text{Softmax}(Q \cdot K_0, \dots, Q \cdot K_n) \quad C = \sum_i \alpha_i \cdot V_i$$

# 交叉注意力 in 中英翻译



设:  $h_1 = [1, 0]$      $h_2 = [0, 1]$      $h_3 = [1, 1]$

注意力权重:  $\alpha_i = \text{Softmax}(\frac{Q_2}{H_2} \cdot \frac{K_i}{h_i})$   
 $\approx [0.27, 0.27, \mathbf{0.46}]$

上下文向量:  $C_2 = \sum_i \alpha_i \cdot V_i = [0.73, 0.73]$

预测输出:  $C_2$  融入  $H_2$  后, 输入分类层, 生成“happy”. 实现了两种语言单词间的软对齐

$$Q_2 \\ H_2 = [0.5, 0.5]$$

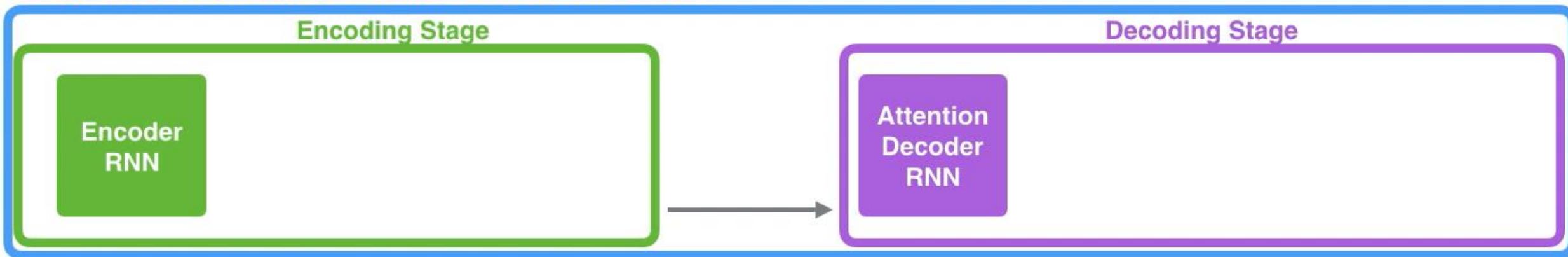
此时相当于查询向量  $Q$   
动态指导注意力的方向

注意力分数  
(点积)

$$\begin{cases} score_1 = H_2 \cdot h_1 = 0.5 \times 1 + 0.5 \times 0 = 0.5 \\ score_2 = H_2 \cdot h_2 = 0.5 \times 0 + 0.5 \times 1 = 0.5 \\ score_3 = H_2 \cdot h_3 = 0.5 \times 1 + 0.5 \times 1 = 1.0 \end{cases}$$

# 带有 注意力

## Neural Machine Translation SEQUENCE TO SEQUENCE MODEL WITH ATTENTION



Je

suis

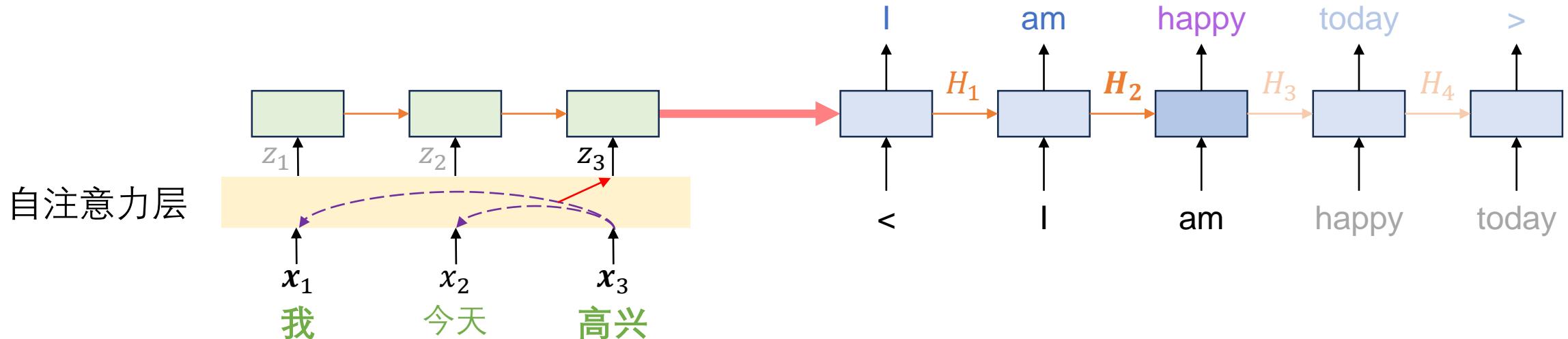
étudiant

# 解码器 在生成时的关注点



# 自注意力

# Self-attention



设:  $x_1 = [1, 0]$      $x_2 = [0, 1]$      $x_3 = \begin{matrix} Q \\ [1, 1] \end{matrix}$

注意力权重:  $\alpha_{3,*} = \text{Softmax}(x_3 \cdot x_i)$   
 $\approx [0.2, 0.2, \mathbf{0.6}]$

上下文向量:  $z_3 = \sum_i \alpha_{3,i} \cdot x_i = \begin{matrix} V_i \\ [0.8, 0.8] \end{matrix}$

预测输出:  $z_3$  融合入“我”、“高兴”的信息, 即表达了“我”、“高兴”的语义关联。

此时相当于查询向量  $Q$ , 关注谁高兴?  
动态指导注意力的方向。

注意力分数  
(点积)

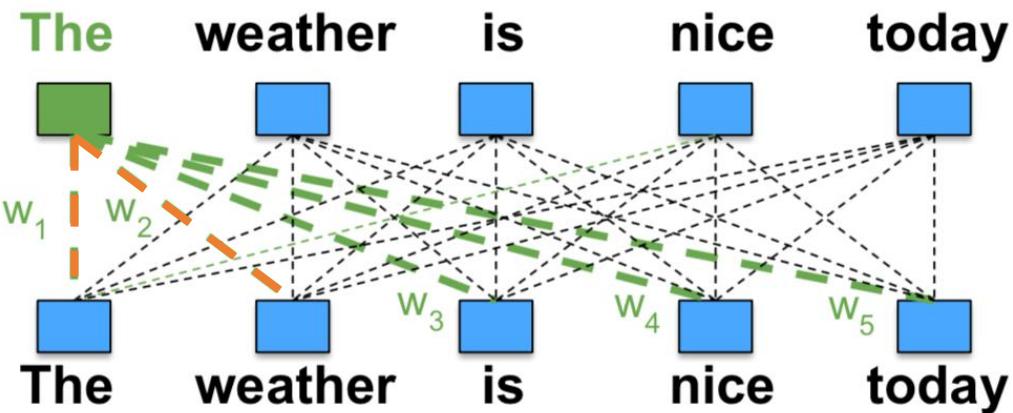
$$\left\{ \begin{array}{l} score_{3,1} = x_3 \cdot x_1 = 1 \times 1 + 1 \times 0 = 1 \\ score_{3,2} = x_3 \cdot x_2 = 1 \times 0 + 1 \times 1 = 1 \\ score_{3,3} = x_3 \cdot x_3 = 1 \times 1 + 1 \times 1 = 2 \end{array} \right.$$

# 自注意力 机制

- 在注意力机制的基础上发展而来
- 处理同一个序列内部各元素间的依赖关系，如，一个句子中的各个词语之间的关系。
- 可以更好地理解上下文信息：无论各元素之间的距离有多远。  
长距离：CNN需要很多层、RNN需要很多时间步。

	交叉注意力	自注意力
输入来源	$Q$ 来自目标序列 $K, V$ 来自 源序列	目标序列 = 源序列
应用场景	处理序列间关系 (如机器翻译)	处理序列内关系 (如文本分类)
目 标	动态对齐不同序列的关联信息	捕捉同一序列内部的上下文依赖关系
数学形式	通用注意力公式	注意力特例 ( $Q, K, V$ 同源)

# 自注意力示例



注意力：从大量信息中有选择地筛选出少量重要信息，并聚焦在其上，忽略大多数不重要信息。

$$w_1, w_2, w_3, w_4, w_5 = \text{softmax} \left( \begin{matrix} 0.6 & 0.2 & 0.8 \\ \text{The} & & \end{matrix} \times \begin{matrix} 0.6 & 0.2 & 0.9 & 0.4 & 0.4 \\ 0.2 & 0.3 & 0.1 & 0.1 & 0.1 \\ 0.8 & 0.1 & 0.8 & 0.4 & 0.6 \\ \text{The} & \text{weather} & \text{is} & \text{nice} & \text{today} \end{matrix} \right)$$
$$\begin{matrix} 1.8 \\ 2.3 \\ 0.4 \end{matrix} = w_1 \times \begin{matrix} 0.6 \\ 0.2 \\ 0.8 \end{matrix} + w_2 \times \begin{matrix} 0.2 \\ 0.3 \\ 0.1 \end{matrix} + w_3 \times \begin{matrix} 0.9 \\ 0.1 \\ 0.8 \end{matrix} + w_4 \times \begin{matrix} 0.4 \\ 0.1 \\ 0.4 \end{matrix} + w_5 \times \begin{matrix} 0.4 \\ 0.1 \\ 0.6 \end{matrix}$$

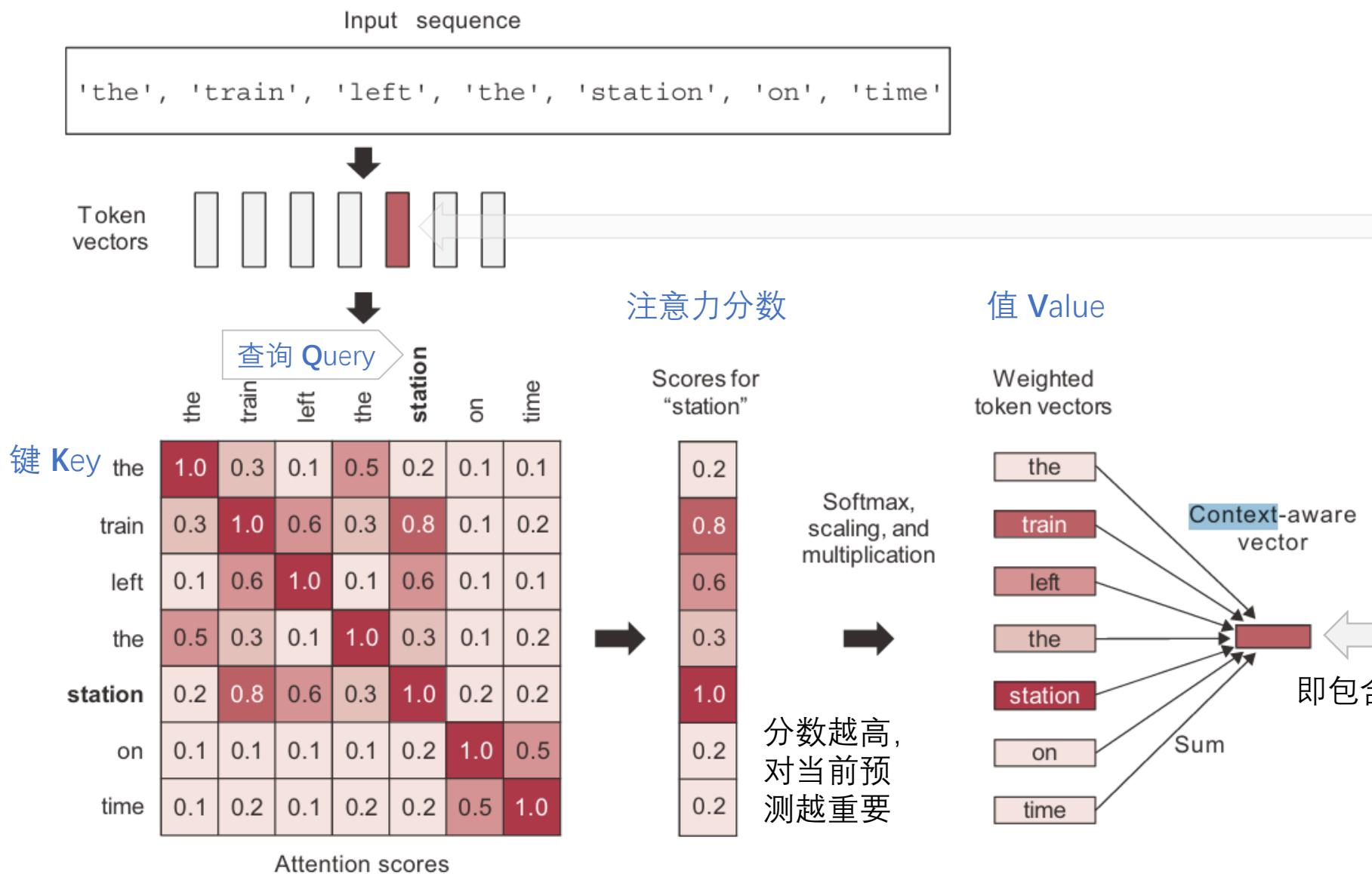
The diagram shows the calculation of attention weights for the word "The". It starts with a query vector [0.6, 0.2, 0.8] for "The" and a key matrix where each row corresponds to a word in the sentence. The key matrix is:

0.6	0.2	0.9	0.4	0.4
0.2	0.3	0.1	0.1	0.1
0.8	0.1	0.8	0.4	0.6
The	weather	is	nice	today

The attention weights are calculated using the softmax function. The resulting weight vector for "The" is [1.8, 2.3, 0.4]. This vector is then multiplied by the query vector to produce the final output vector.

聚焦体现在权重系数上，权重越大，与 The 联系越紧密。

# 自注意力 计算



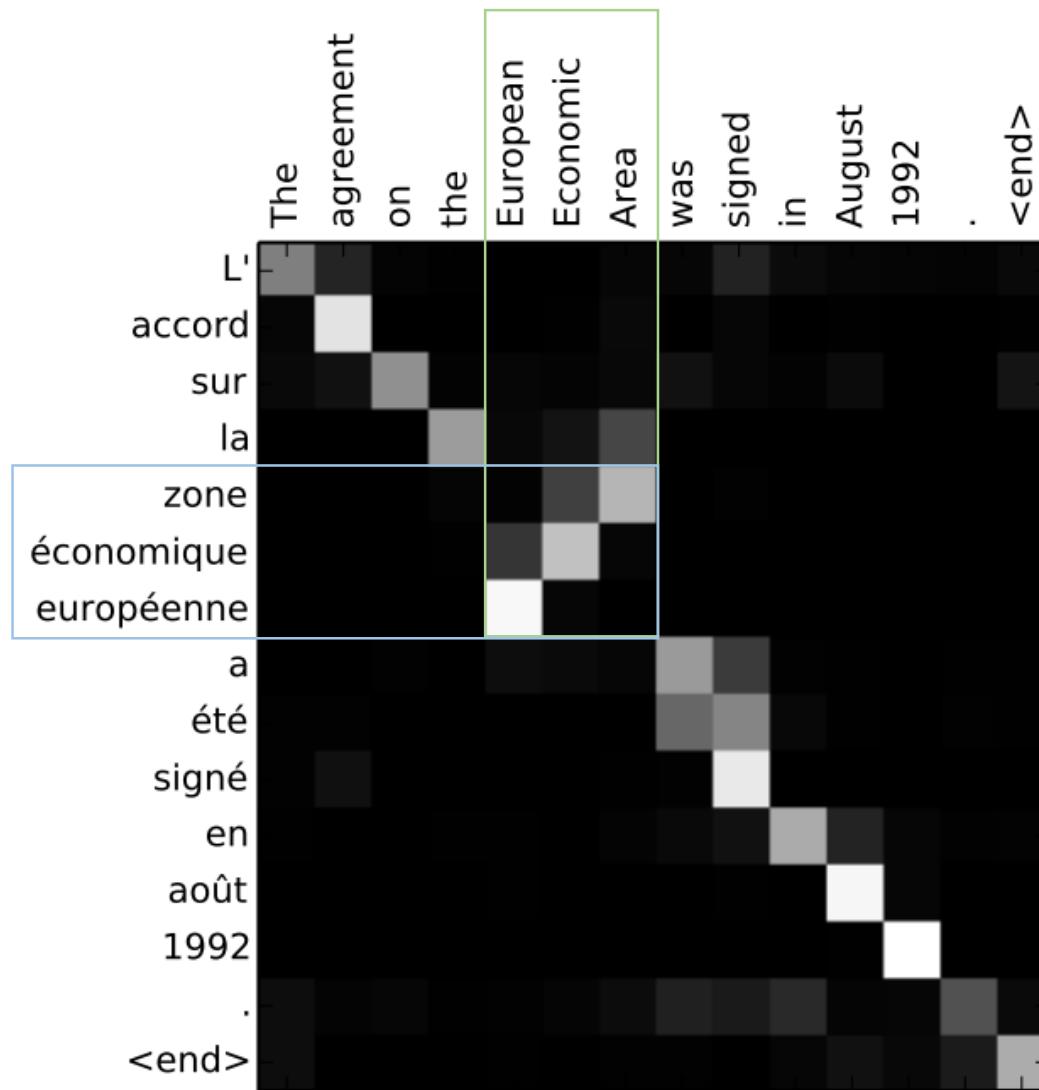
词向量 旧表示

- 单词 station 需要去查询 Q 句子中所有单词,
- 所有单词需要有键 Key 值,
- 查出来的 注意力分数 \* 所有单词向量 Value = 所有单词新的向量值。
- 求和 = station 的新向量

词向量 新表示, 含上下文

即包含了 train 向量部分, 表明 train station.

# 自注意力 可视化

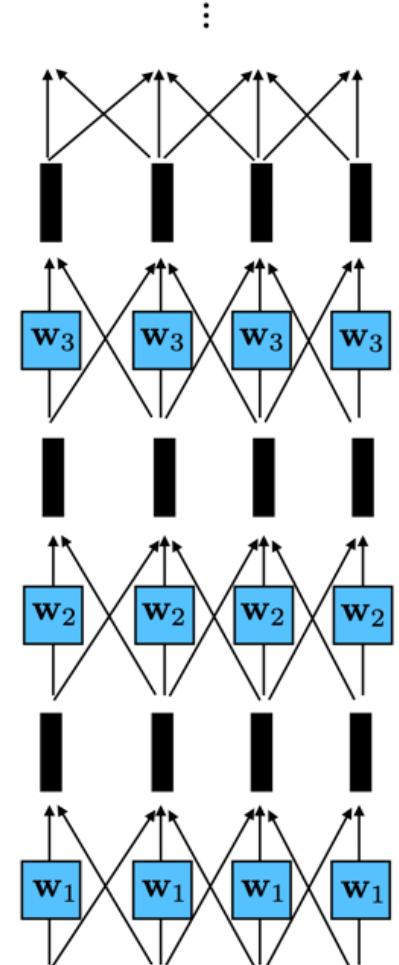


输出英文 “European Economic Area”

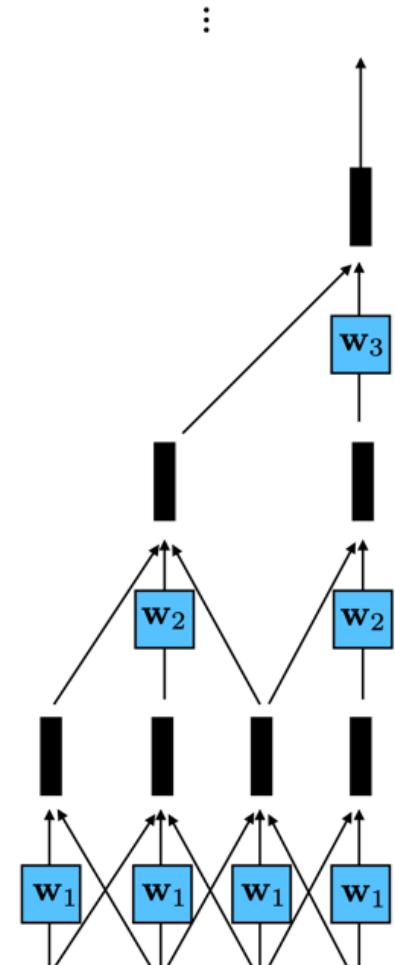
与法语中 “zone économique européenne”  
对应单词的顺序是相反的。

# 卷积层

conv w overlap

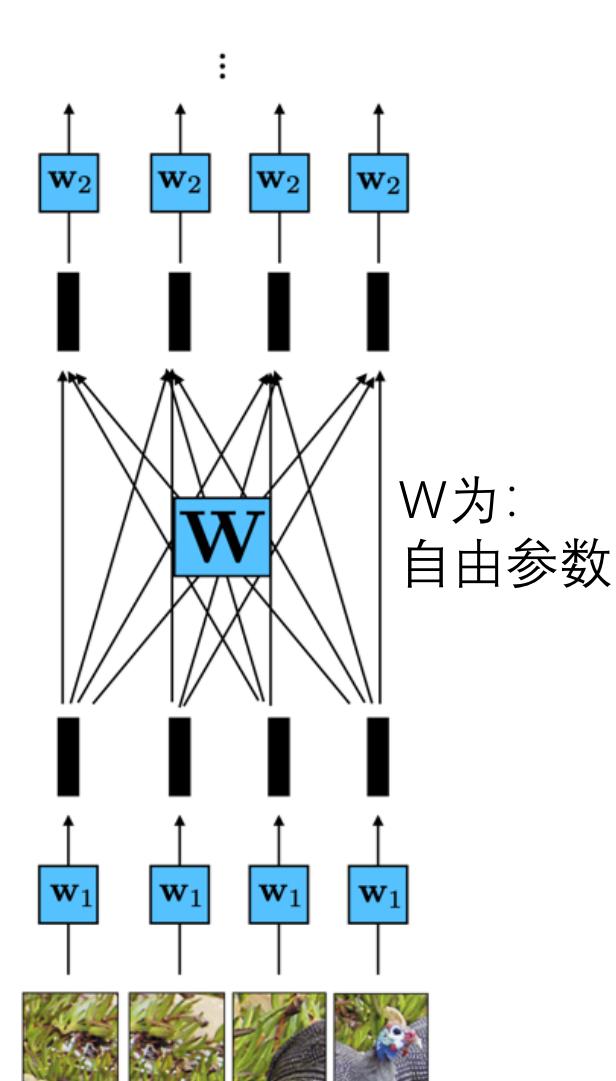


conv pyramid



# 全连接层

fc layer



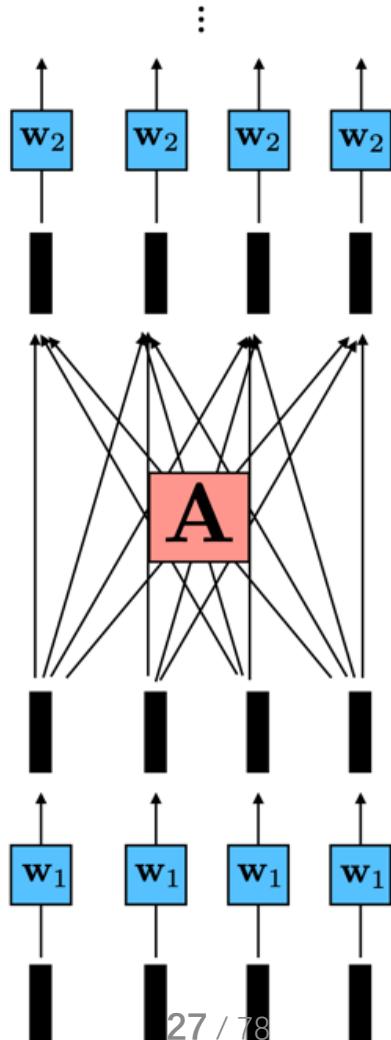
# 自注意力层

$$Z_{out} = \mathbf{A} Z_{in}$$

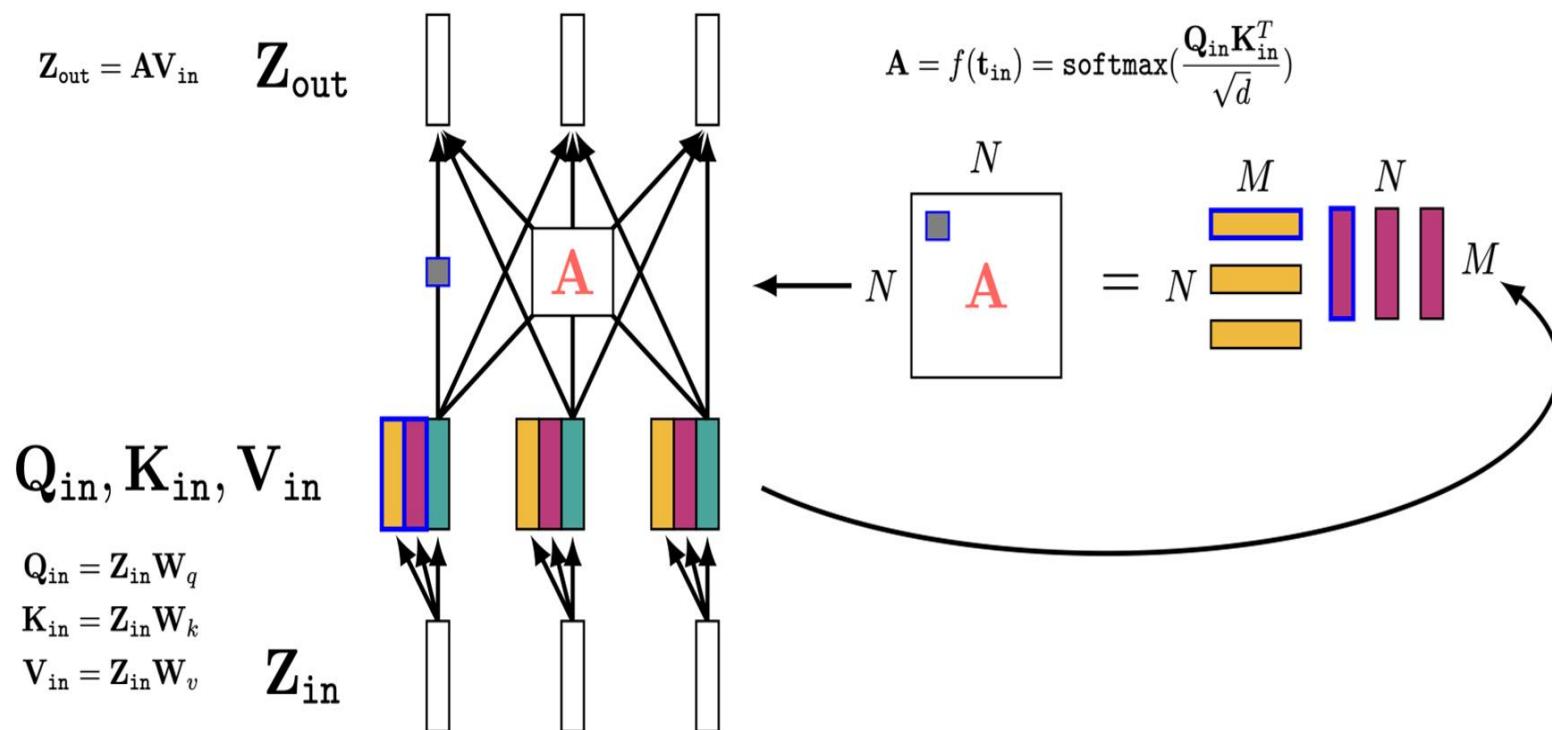
$$\mathbf{A} = f(\dots)$$

一些数据的函数  $f$

这些数据提示哪些词元应该多注意。



# 自注意力层



自注意力计算步骤：

- ① 将输入表示为特征向量  $Z_{in}$
- ② 输入矩阵  $Z_{in}$  变换为查询矩阵 (Q)、键矩阵 (K)、值矩阵 (V)
- ③ 计算注意力权重矩阵 (A)
- ④ 计算输出矩阵  $Z_{out} = A \cdot V_{in}$

# 3

# Transformer

Google: *Attention Is All You Need*, Computation and Language, 2017, 2023-v7

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin,

加入  
**DeepMind**  
强化学习

创办  
**Character.AI**  
个性化对话AI

共同创立  
**Inceptive**  
AI算法落地、AI设计RNA

加入  
**Sakana AI**  
探索新AI架构

共同创立  
**Cohere**  
大语言模型

加入  
**Anthropic**  
AI 安全

联合创立  
**NEAR Protocol**  
区块链

# Transformer

- 改变了自然语言**理解**的游戏规则,
- **革命性**、**颠覆性**的，将CNN、RNN(需大算力)甩在身后。
- **没有循环**，没有CNN，没有RNN，也没有LSTM
- 注意力机制是“**单词对单词**”，可发现**每个单词与序列中所有单词间的关联关系**！  
全局视角：**长距离依赖问题**。
- **并行计算**

原始Transformer模型在450万句子对英语-德语 + 3,600万句英语-法语数据集上训练  
8个英伟达 P100 GPU，基础模型训练12小时(10万步)，大模型3.5天(30万步)  
41.8 BLEU 评分优于以前所有的机器翻译模型。

# 双语评估辅助工具 BLEU

Bilingual Evaluation Understudy

- 常用的机器翻译质量评估指标，评估机器翻译输出的质量。
- 通过计算候选翻译与一个或多个参考翻译之间的 n-gram (连续的n项序列)重叠程度来衡量翻译的质量，分数越高翻译质量越好。

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	<b>41.29</b>	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1		<b><math>3.3 \cdot 10^{18}</math></b>
Transformer (big)	<b>28.4</b>	<b>41.8</b>		$2.3 \cdot 10^{19}$

# BLEU 指标 Bilingual Evaluation Understudy

通过比较候选翻译(模型输出)、参考翻译(人工翻译)之间的 n-gram 重叠程度来打分。

候选1-gram : ["男孩", "踢", "足球"] # 候选长度  $c = 5$

参考1-gram : ["一个", "小", "男孩", "在", "踢", "足球"] # 参考长度  $r = 9$

候选2-gram : ["男孩 踢", "踢 足球"]

参考2-gram : ["一个 小", "小 男孩", "男孩 在", "在 踢", "踢 足球"]

候选3-gram : ["男孩 踢 足球"]

参考3-gram : ["一个 小 男孩", "小 男孩 在", "男孩 在 踢", "在 踢 足球"]

候选4-gram : 无 (候选长度=3, 无法生成4-gram)

参考4-gram : ["一个 小 男孩 在", "小 男孩 在 踢", "男孩 在 踢 足球"]

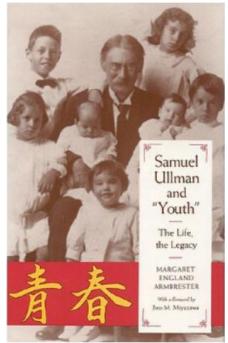
句子长度惩罚因子: 防止过短的候选翻译得分虚高。  $BP = e^{(1-9/5)} \approx 0.4493$

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases}$$

$$\text{BLEU} = BP \times \exp\left(\sum_{n=1}^4 w_n \log p_n\right) = 0.4493 \times \exp(0.25 \log 1 + 0.25 \log 0.5 + 0.25 \log 0 + 0.25 \log 0) \approx 0.4167$$

$w_n$ : 各n-gram的权重(默认均匀权重)

# 机器翻译 现状



美国作家 Samuel Ullman, *Youth* 原文:

Youth is not a time of life, it is a state of mind; it is not a matter of rosy cheeks, red lips and supple knees; it is a matter of the will, a quality of the imagination, a vigor of the emotions; it is the freshness of the deep springs of life.

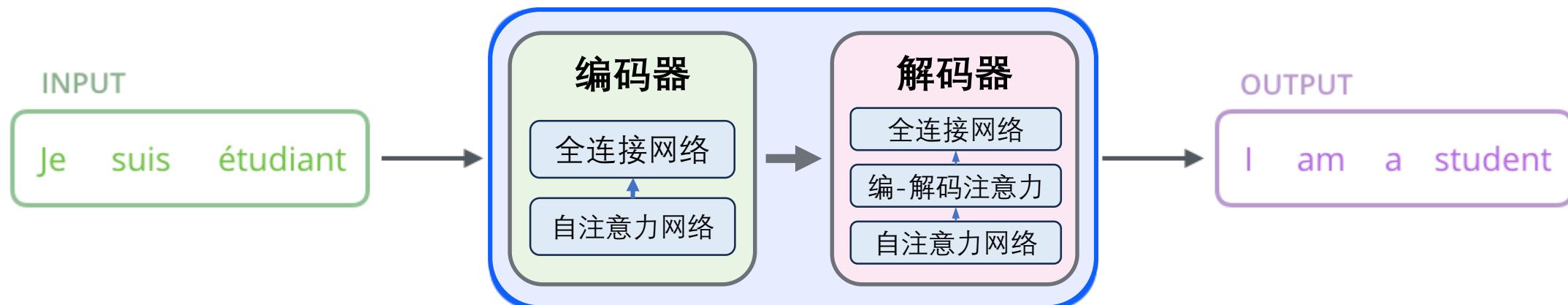
**王佐良译文:** 青春不是年华，而是心境；青春不是桃面、丹唇、柔膝，而是深沉的意志，恢宏的想象，炙热的感情；青春是生命的深泉在涌流。

**DeepSeek :** 青春不是年华，而是心境；青春不是桃面、丹唇、柔膝，而是深沉的意志、恢宏的想象、炽热的情感；青春是生命深泉的自在奔流。

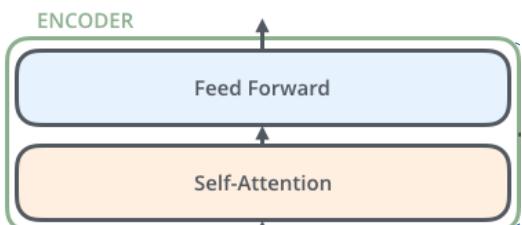
**Google翻译:** 青春不是人生的一个阶段，而是一种心境；青春不是桃面、丹唇、柔膝；青春是一种意志，一种想象，一种情感的活力；青春是生命深泉的清新。

**百度翻译:** 青春不是一段时间，而是一种心态；这不是玫瑰色的脸颊、红唇和柔软的膝盖；它是一种意志，一种想象力，一种情感的活力；它是生命深泉的清新。

# 解析 Transformer



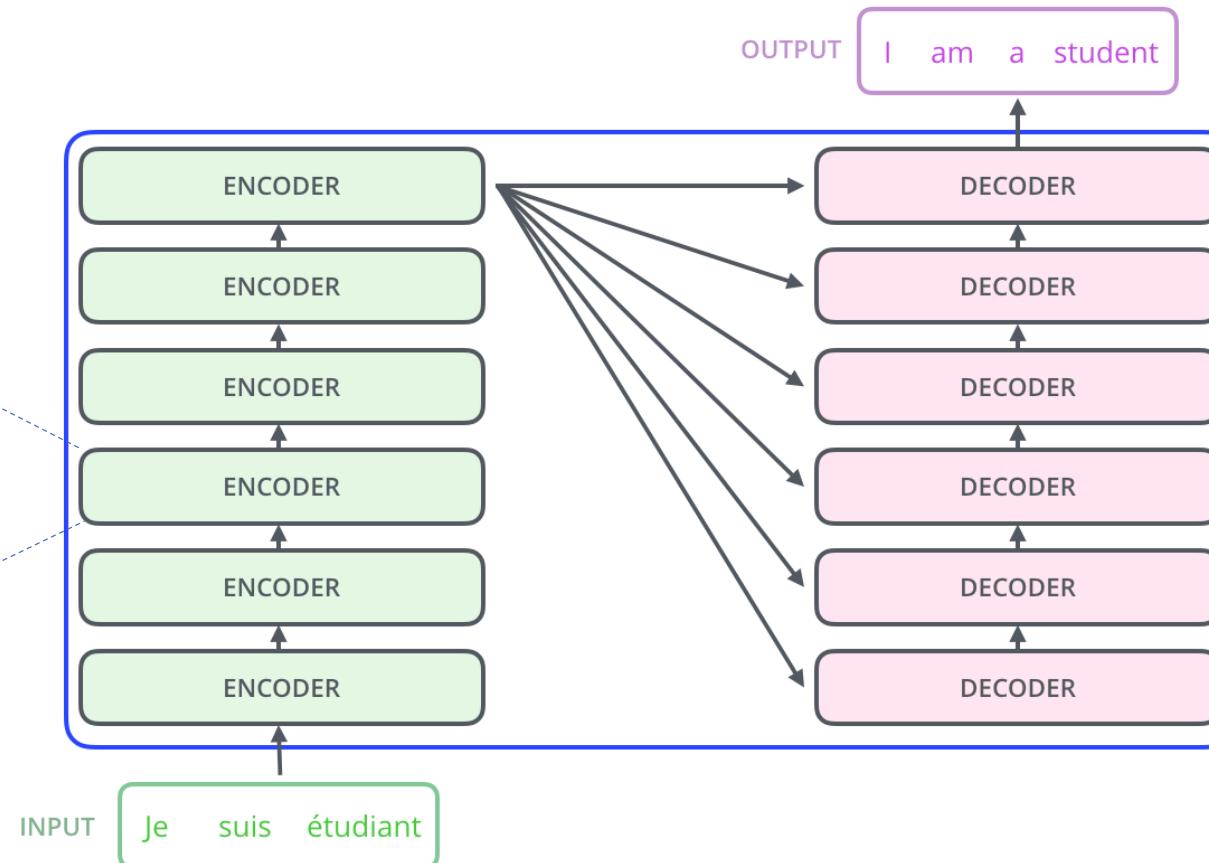
# 解析 Transformer：编码器、解码器



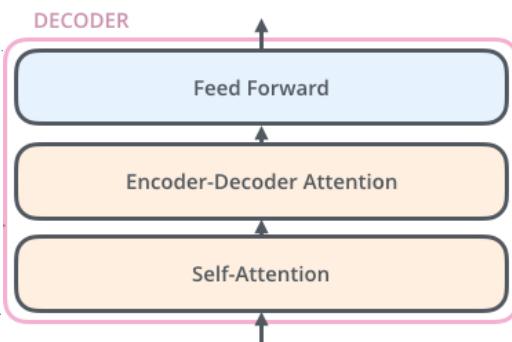
**自注意力层：**

让特定单词，查看输入  
句子中的其他单词。

即，自我关注



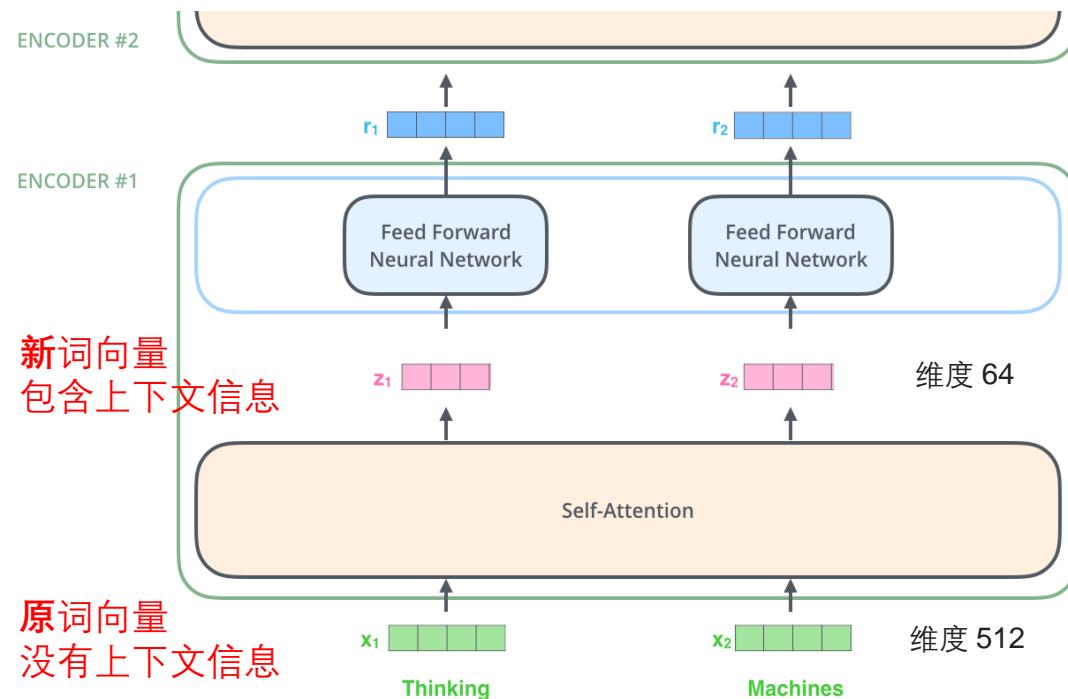
**编码-解码交叉注意力层：**  
让解码器专注于输入句子的  
相关部分。



**解码器：**

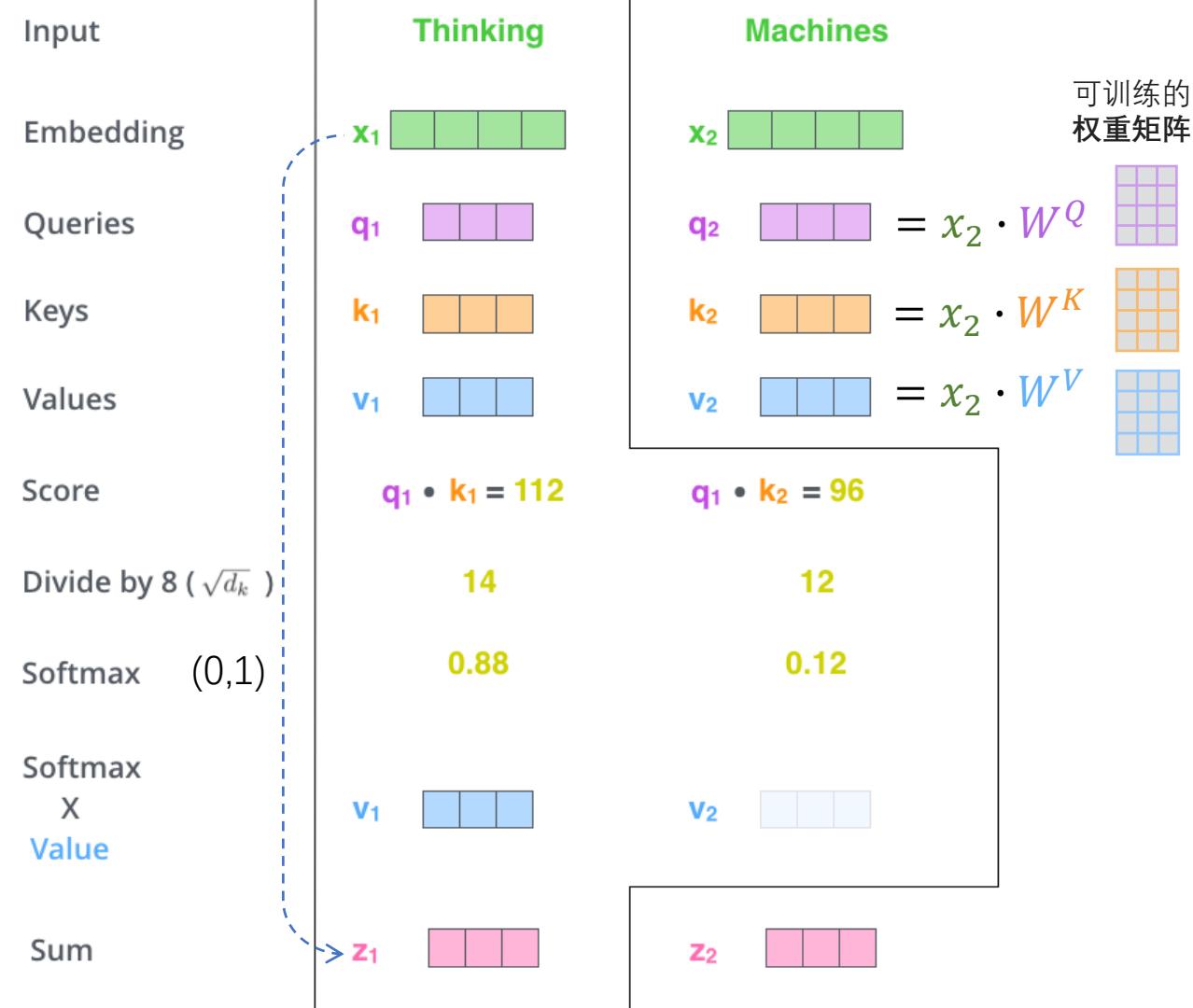
既要关注输入的单词，  
还要关注已翻译好的单词。

# 解析 Transformer: 编码器



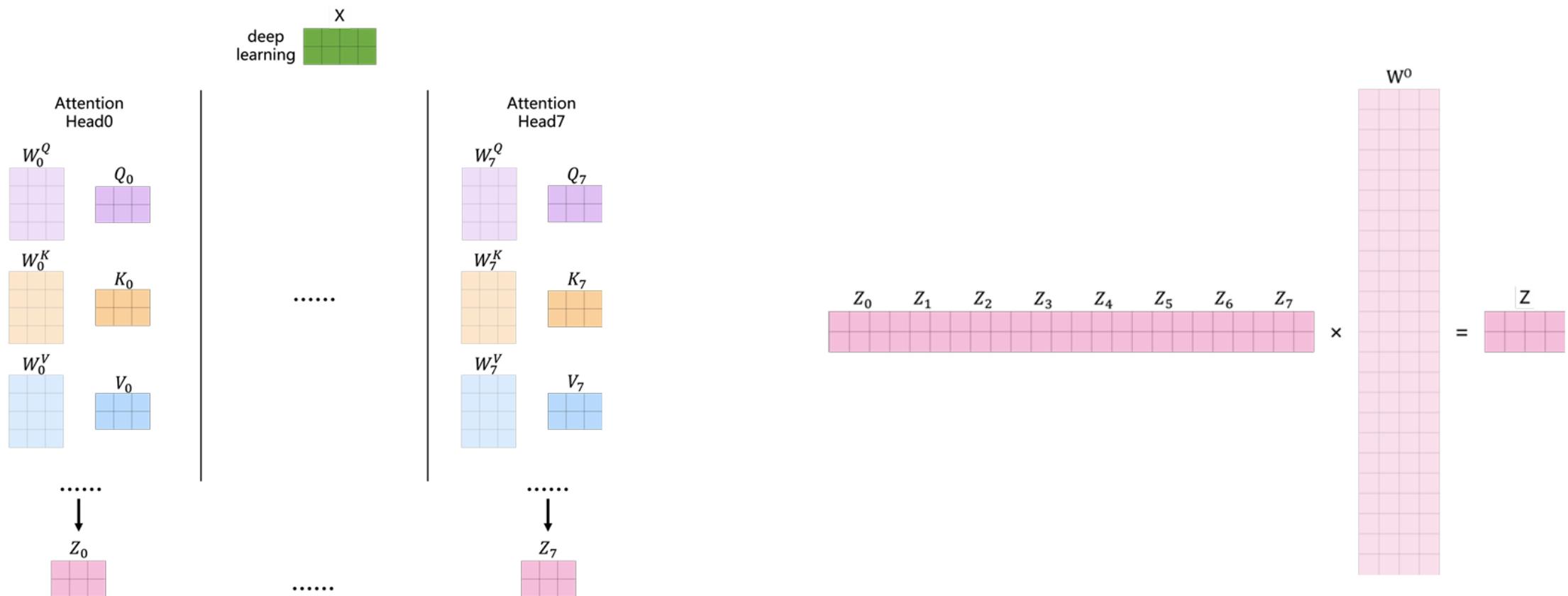
$$\begin{aligned} x &\times W^Q = Q \\ x &\times W^K = K \\ x &\times W^V = V \end{aligned}$$

$$Z = \text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) V$$



# 解析 Transformer: 多头

- “多头”注意力：提供了多个“表示子空间”。
- 每组权值将输入嵌入投影到不同的表示子空间中。

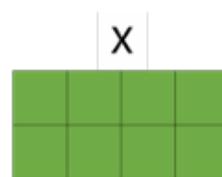


# Multi-Head Attention 计算流程

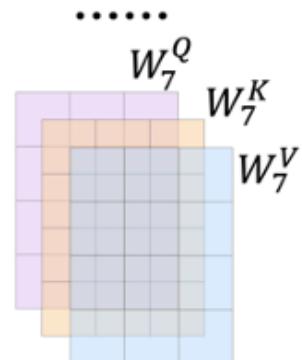
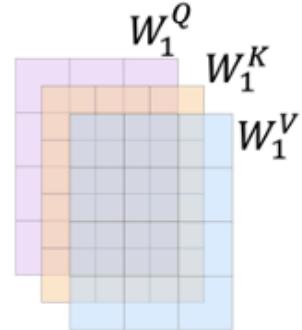
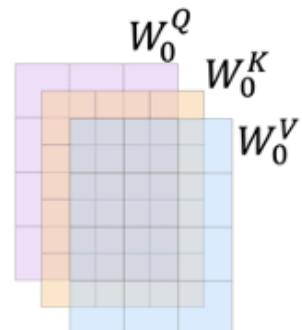
1.输入句子

deep learning

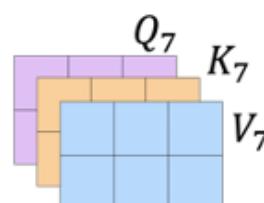
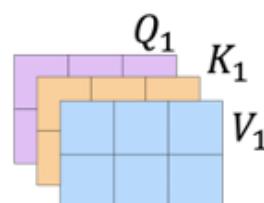
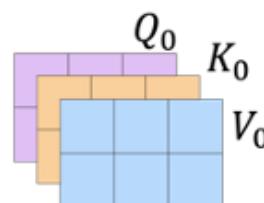
2.把词变成向量



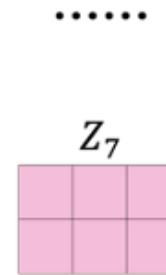
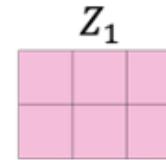
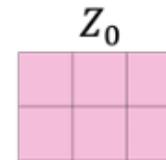
3.分别乘以8个Attention Head中的 $W^Q$ ,  $W^K$ ,  $W^V$



4.使用Q/K/V矩阵来计算attention



5.计算 $Z_0$ 到 $Z_7$

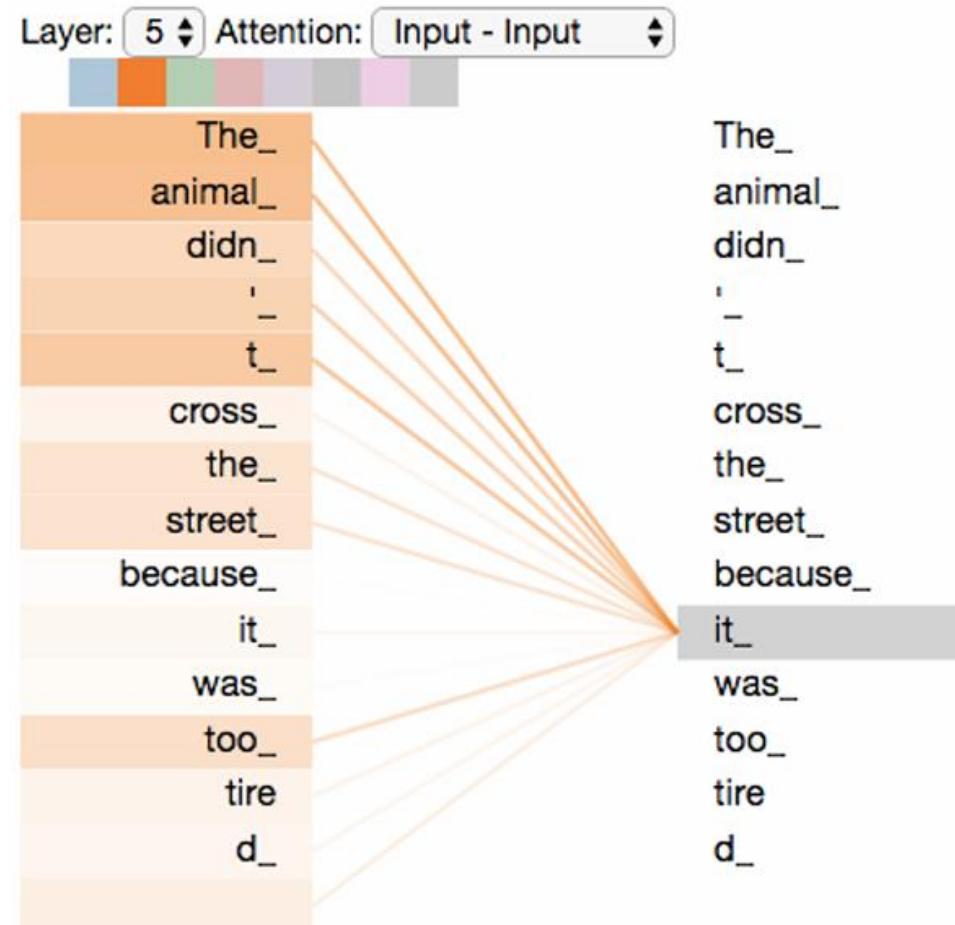


6.合并 $Z_0$ 到 $Z_7$ 乘以 $W^O$ 得到最终输出结果Z



# 自注意力 可视化

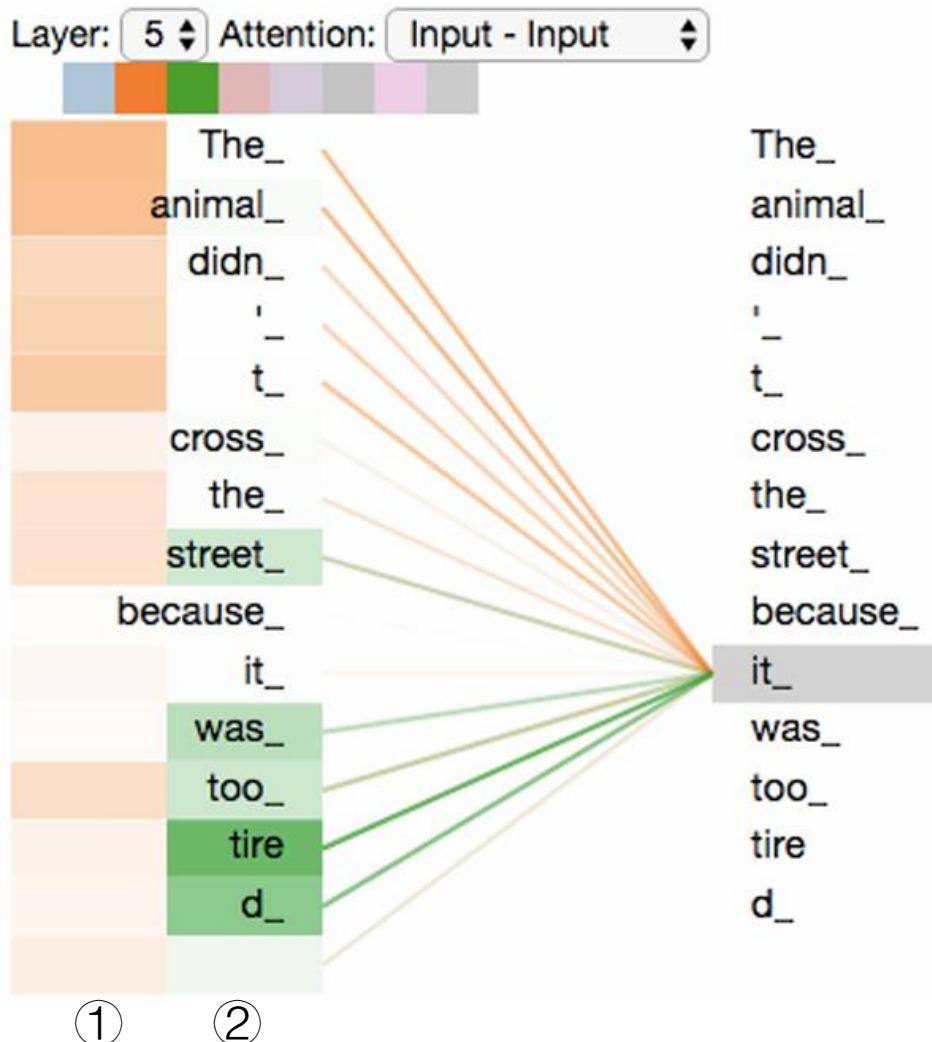
"The **animal** didn't cross the street because **it** was too tired."



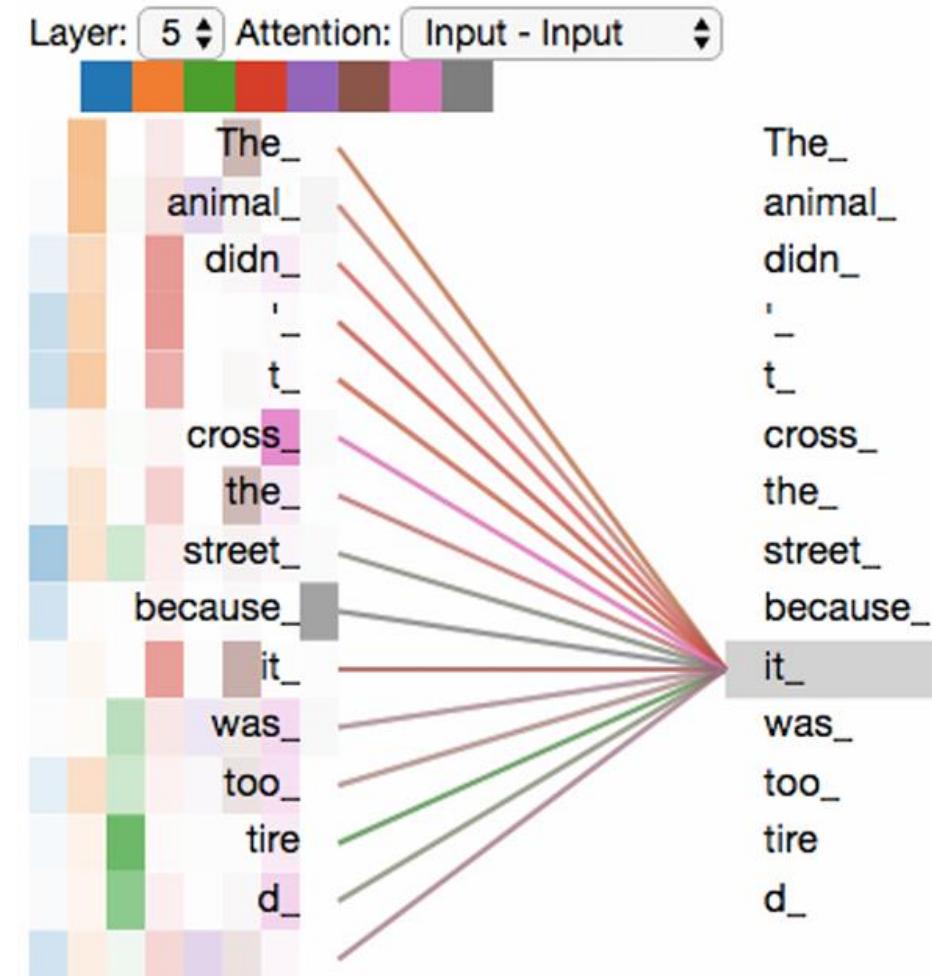
Self-Attention让机器把  
“it”和“animal”联系起来。

# 多头自注意力 可视化

2 头关注句子中不同的部位

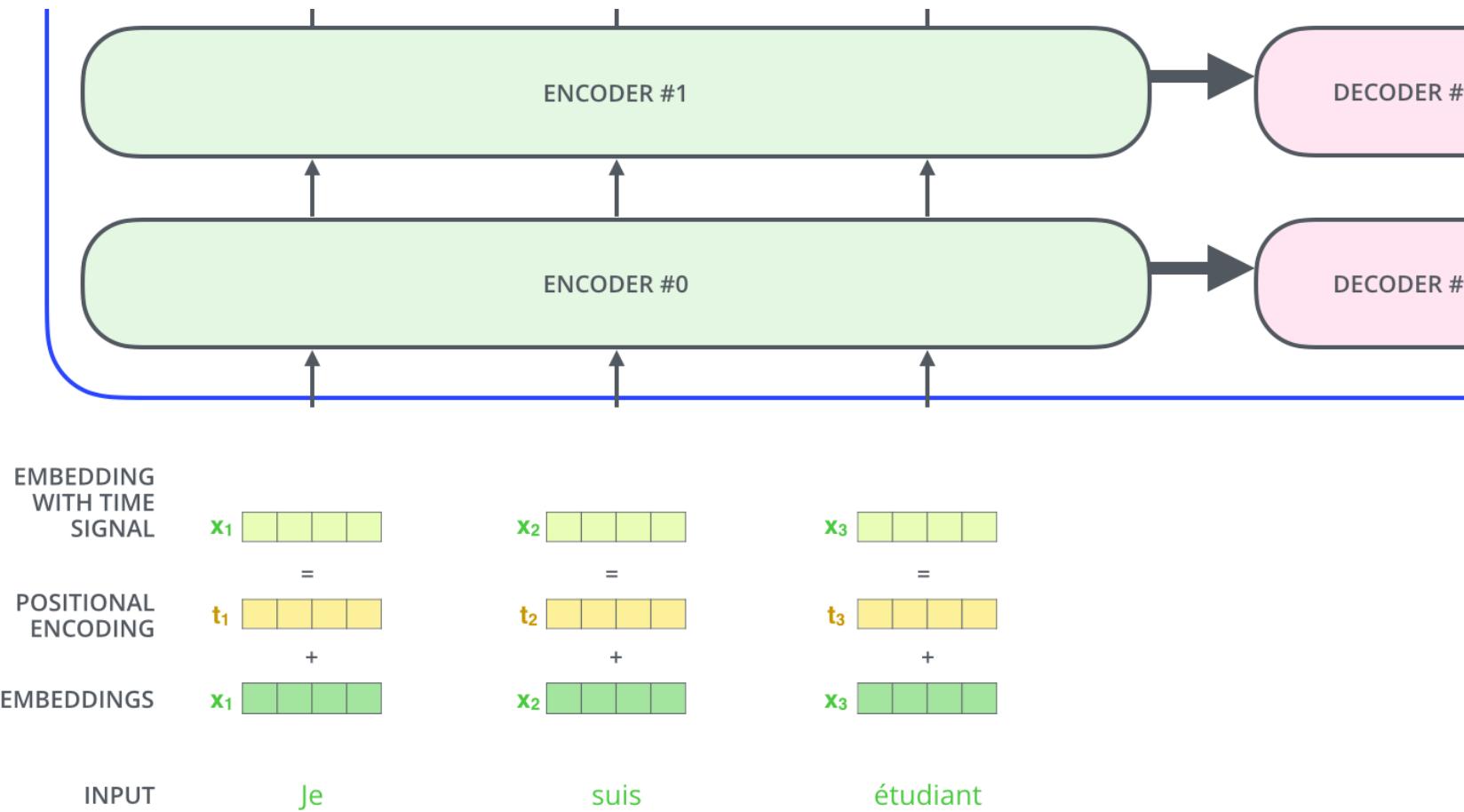


8 个头结果:



# 解析 Transformer: 位置信息

- Transformer本身没有位置信息，位置信息是输入时加入的。
- 位置信息表达方式：绝对位置信息、相对位置信息。



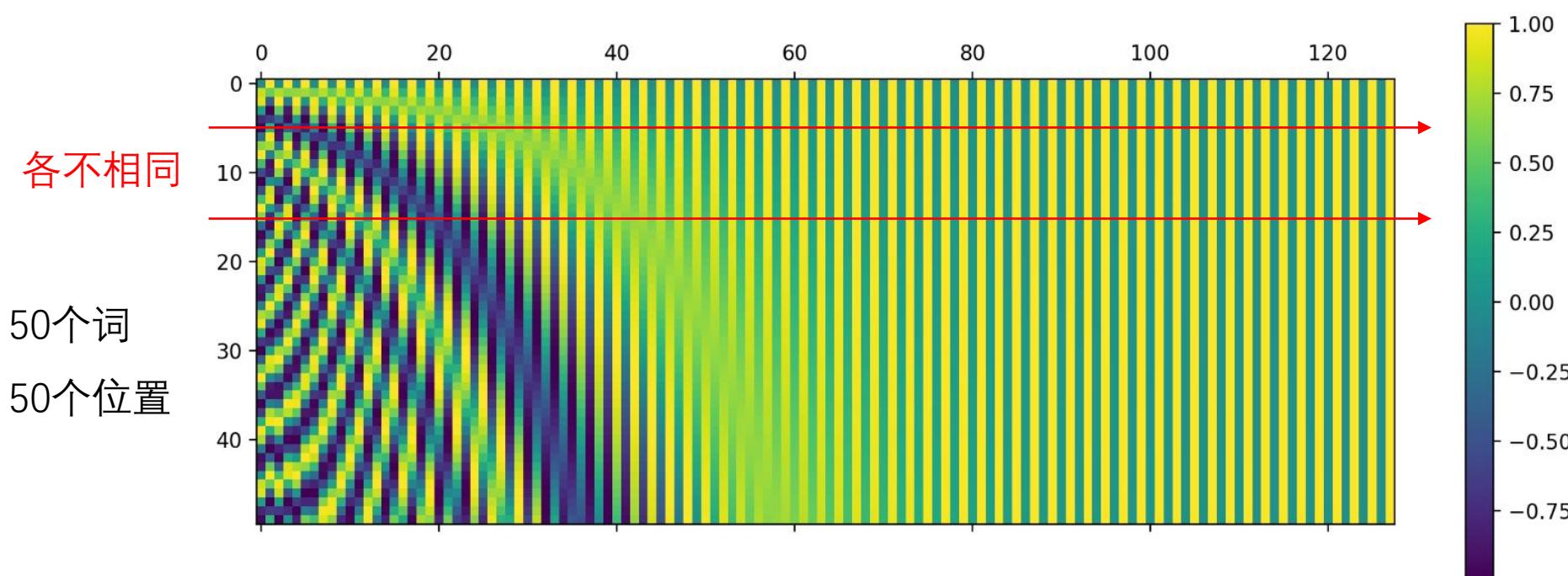
# 相对位置

$$\text{PE}_{(pos, 2i)} = \sin\left(\frac{pos}{10000^{2i/d}}\right)$$

$$\text{PE}_{(pos, 2i+1)} = \cos\left(\frac{pos}{10000^{2i/d}}\right)$$

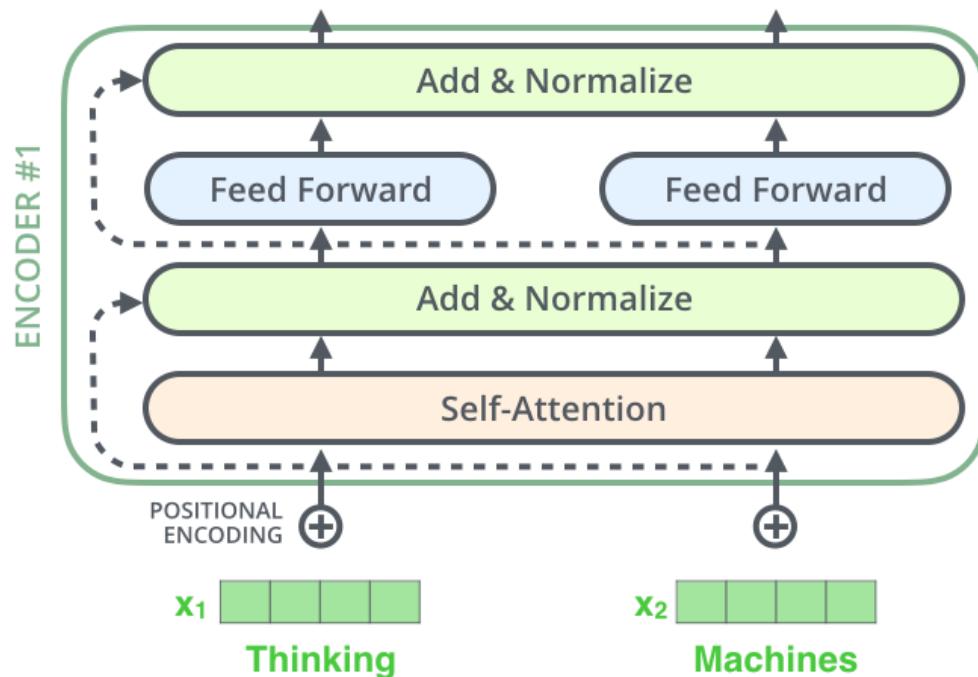
$pos$ : 当前词在句中的位置  
如  $pos_{max} = 50$

$d$ : 词向量长度, 如  $= 128$

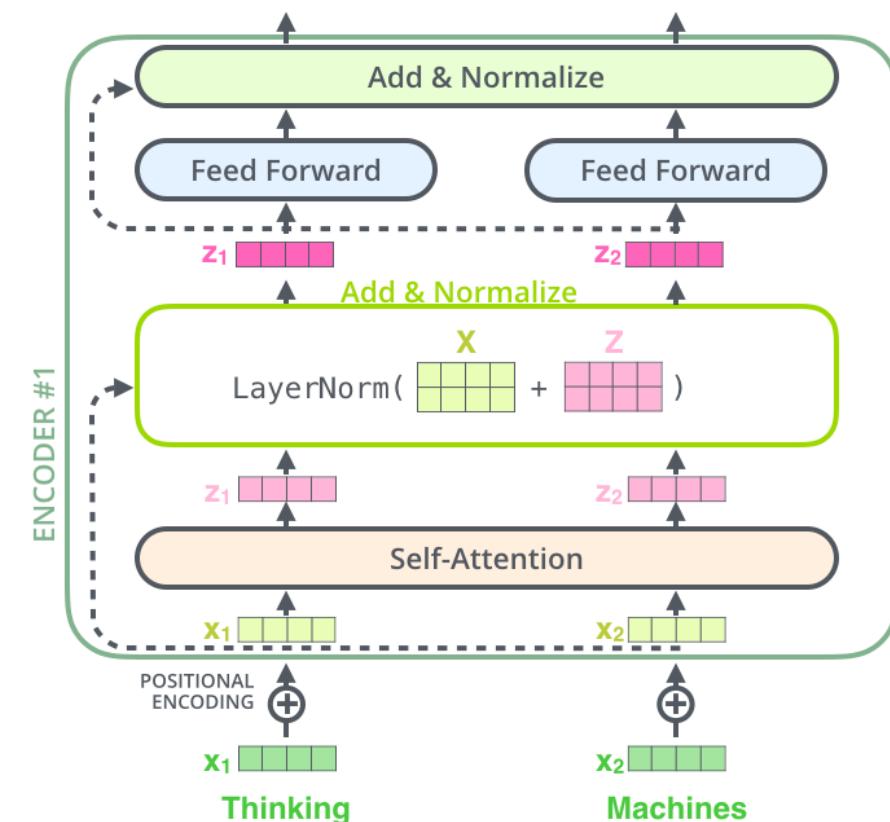


# 解析 Transformer: 编码器

残差连接



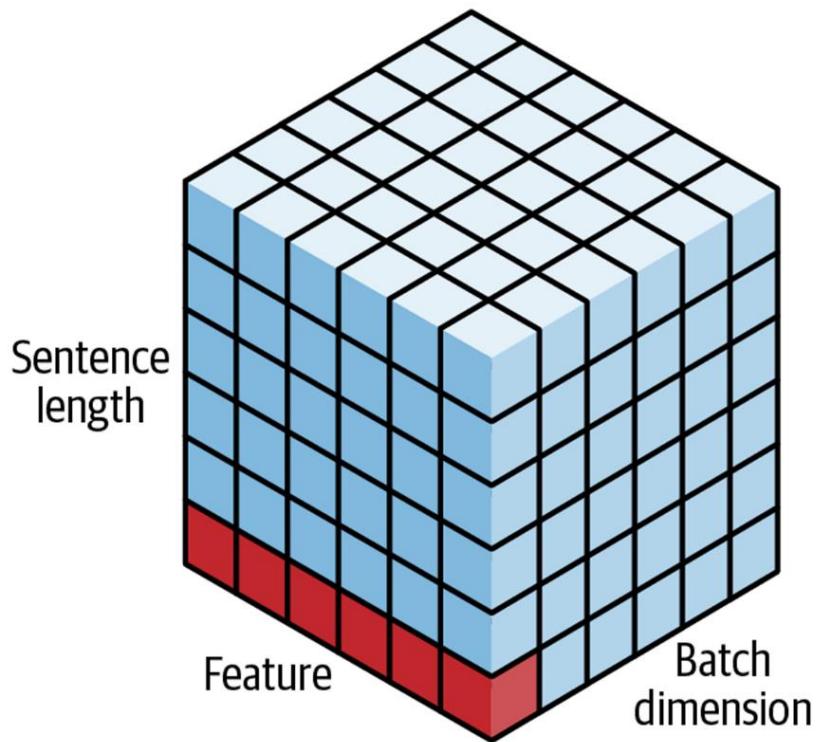
层归一化



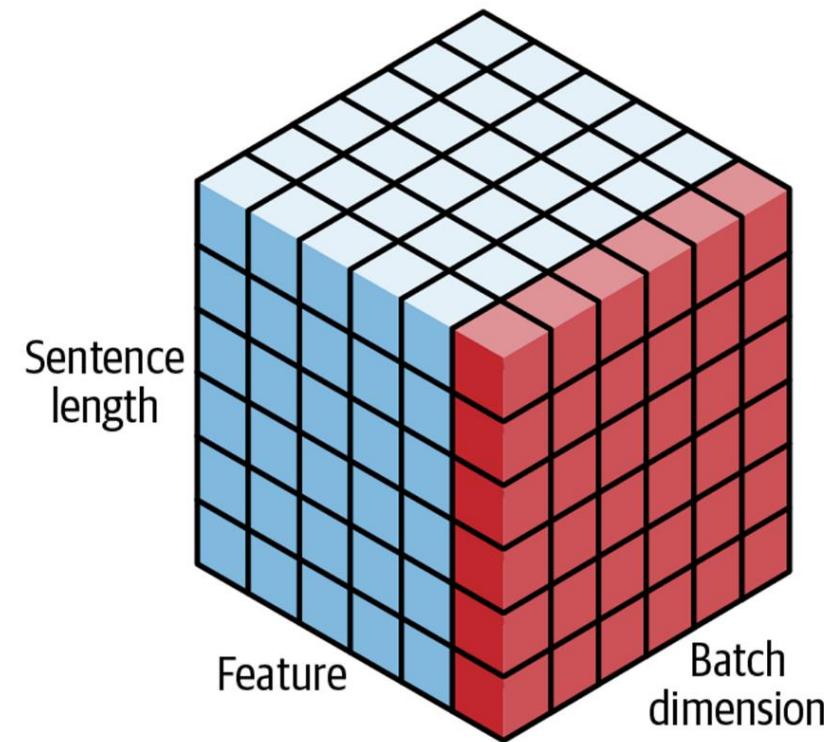
# Layer Normalization

提升模型训练的稳定性、收敛速度。

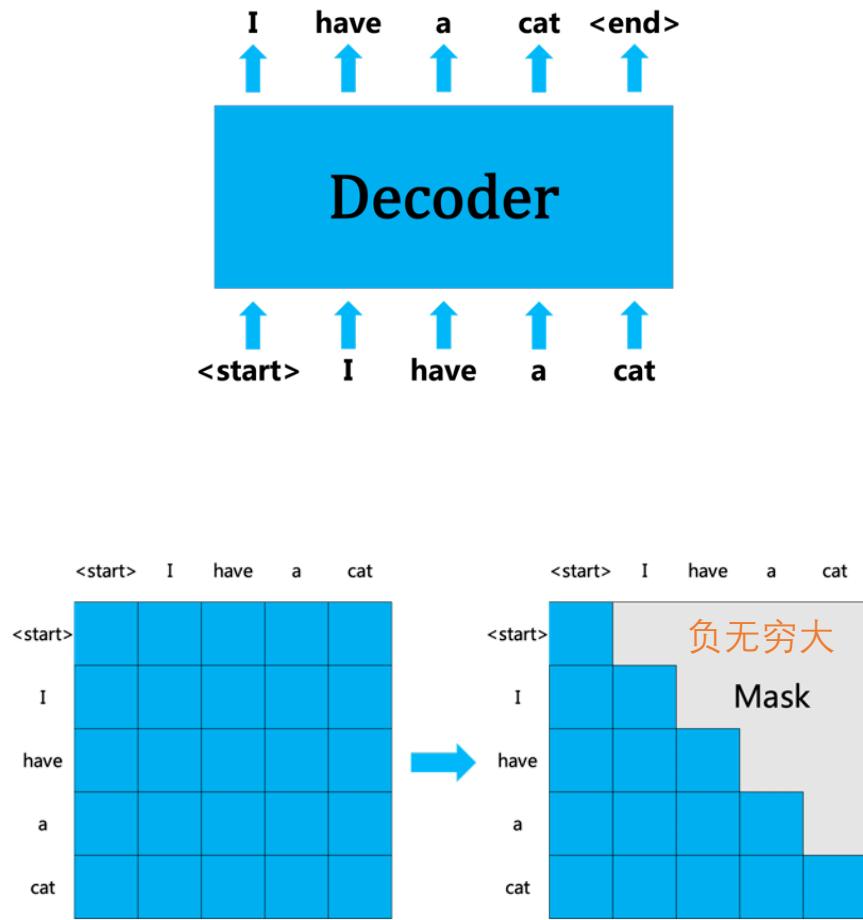
Layer normalization



Batch normalization



# Mask 操作



$$Q \times K^T = QK^T$$

A diagram showing the multiplication of two matrices. On the left is a purple 5x5 matrix labeled  $Q$ . In the middle is an orange 5x5 matrix labeled  $K^T$ . To the right is a blue 5x5 matrix labeled  $QK^T$ . Between the  $Q$  and  $K^T$  matrices is a multiplication symbol ( $\times$ ). To the right of the  $QK^T$  matrix is an equals sign (=).

$$\text{Element-wise product} \times \text{Mask} = \text{Mask } QK^T$$

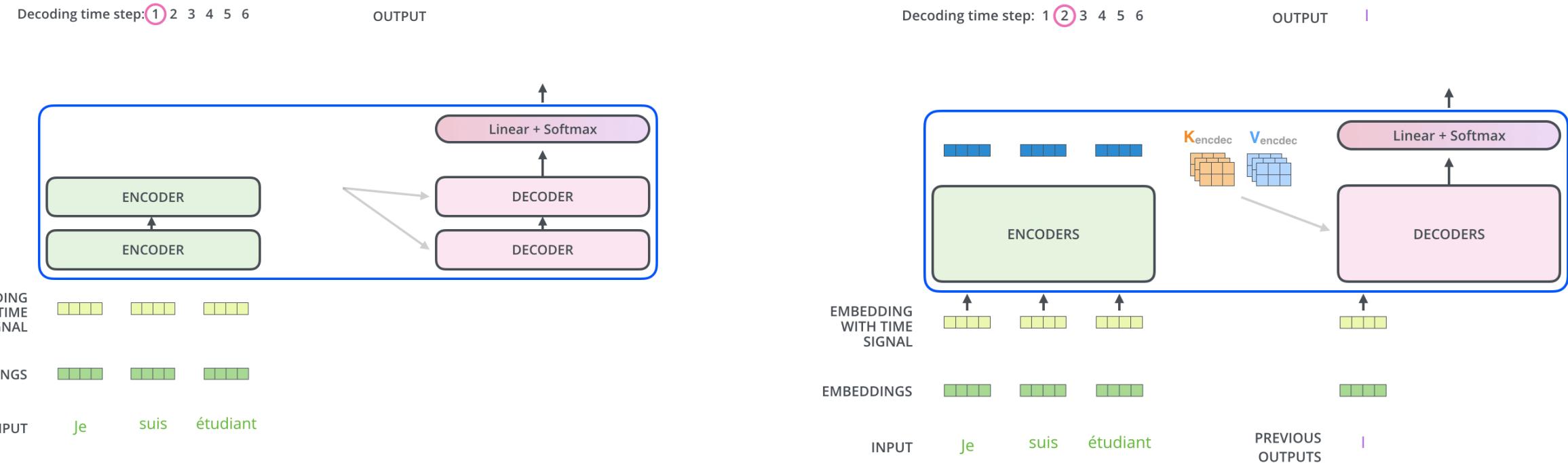
A diagram showing the element-wise product of the  $QK^T$  matrix and a mask matrix. On the left is a blue 5x5 matrix labeled  $QK^T$ . In the middle is a blue 5x5 matrix labeled "Mask". To the right is a blue 5x5 matrix labeled  $\text{Mask } QK^T$ . Between the  $QK^T$  and "Mask" matrices is an "Element-wise product" symbol. To the right of the  $\text{Mask } QK^T$  matrix is an equals sign (=).

$$\text{softmax}(\text{Mask } QK^T) \times V = Z$$

A diagram showing the softmax operation followed by a multiplication. On the left is a blue 5x5 matrix labeled "Mask". In the middle is a blue 5x5 matrix labeled  $V$ . To the right is a pink 5x5 matrix labeled  $Z$ . Between the "Mask" and  $V$  matrices is a softmax symbol ( $\text{softmax}$ ). To the right of the  $Z$  matrix is an equals sign (=).

只包含单词1的信息  
包含单词1和单词2信息  
.....  
.....  
.....

# 解析 Transformer: 编码器 → 解码器



- **编码器**先处理输入序列。然后，在编码器顶部的输出被转换为一组注意力向量  $K$  和  $V$ 。
- 每个解码器在“encoder-decoder attention”层使用  $K$ 、 $V$ ，可专注于 input 序列中的适当位置。

- **解码器**: self-attention 层只允许关注输出序列中的较早位置 (mask)。

# 解析 Transformer: 解码器输出

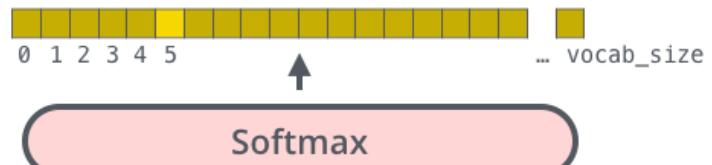
从词典中选出词

am

最大概率的编号

5

概率最大值



logits 向量

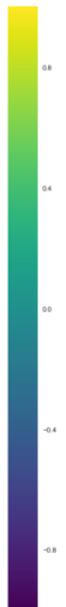


解码器输出值

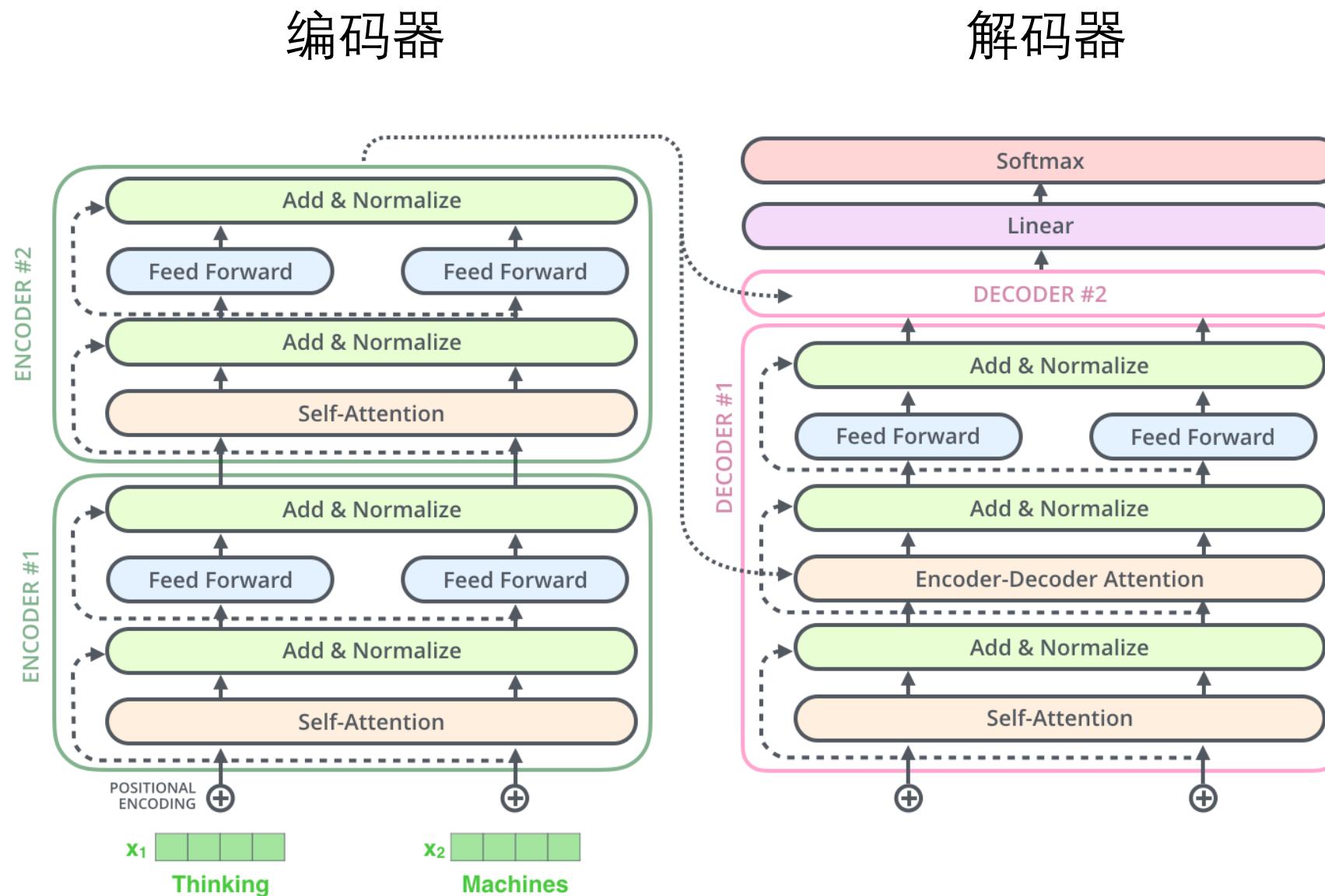
Trained Model Outputs

假设词典中单词数量 = 6

	a	am	I	thanks	student	<eos>
1	1	2	3	4	5	6
position #1	0.01	0.02	0.93	0.01	0.03	0.01
position #2	0.01	0.8	0.1	0.05	0.01	0.03
position #3	0.99	0.001	0.001	0.001	0.002	0.001
position #4	0.001	0.002	0.001	0.02	0.94	0.01
position #5	0.01	0.01	0.001	0.001	0.001	0.98

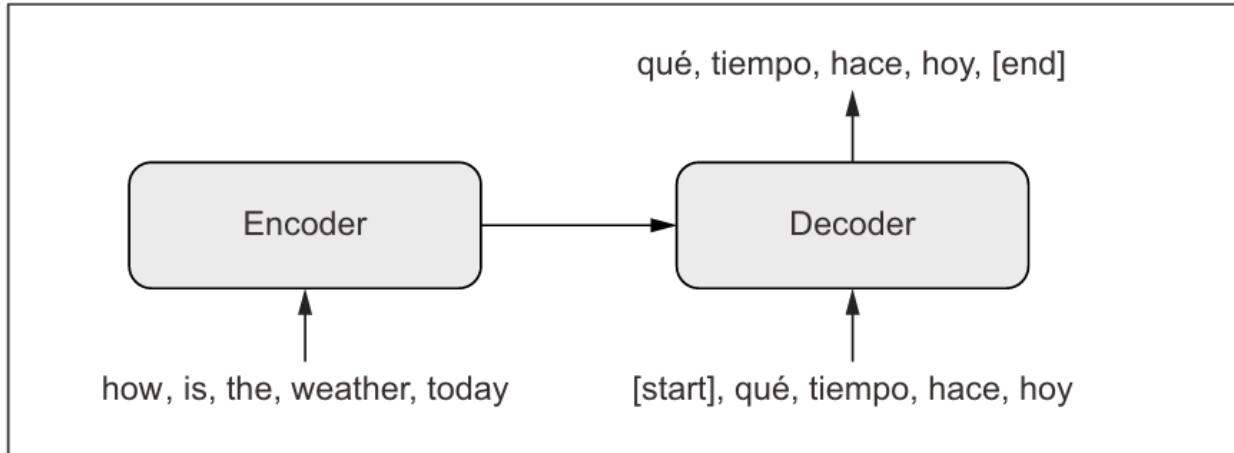


# 解析 Transformer: 编码器 + 解码器

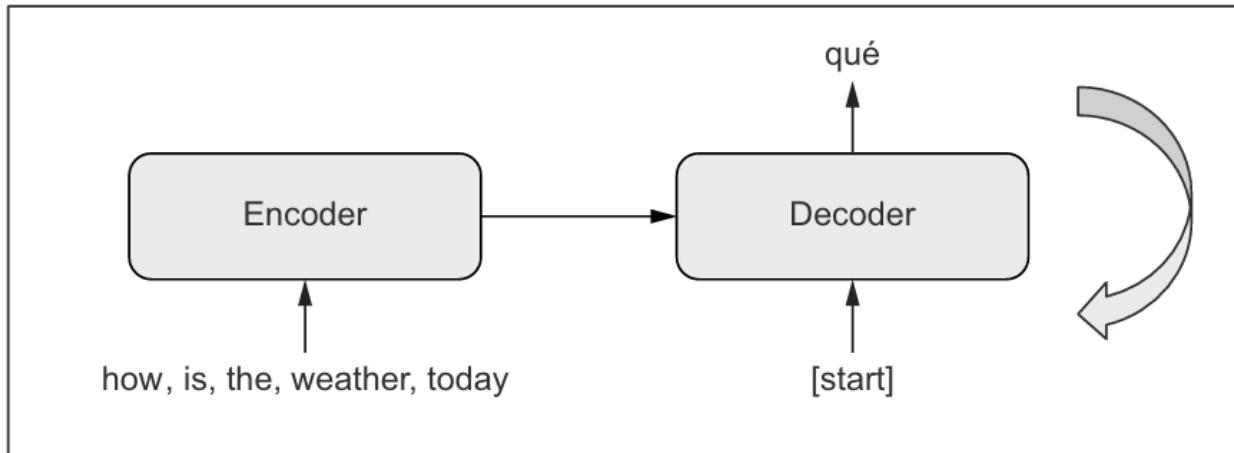


# 训练、预测

训练



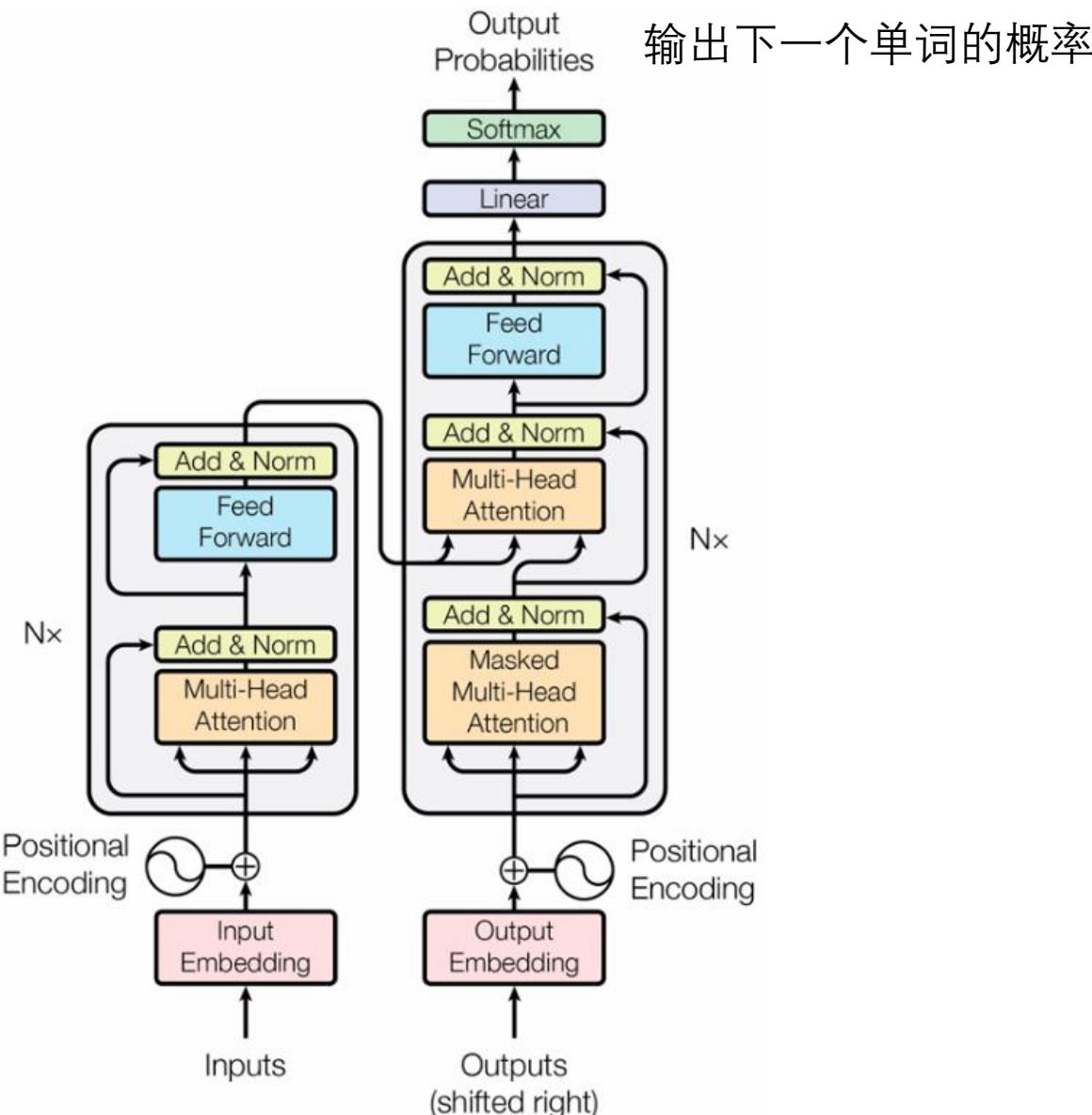
预测



并行  
用Mask

串行  
不用Mask

# Transformer 流程图



# Transformer 英译法

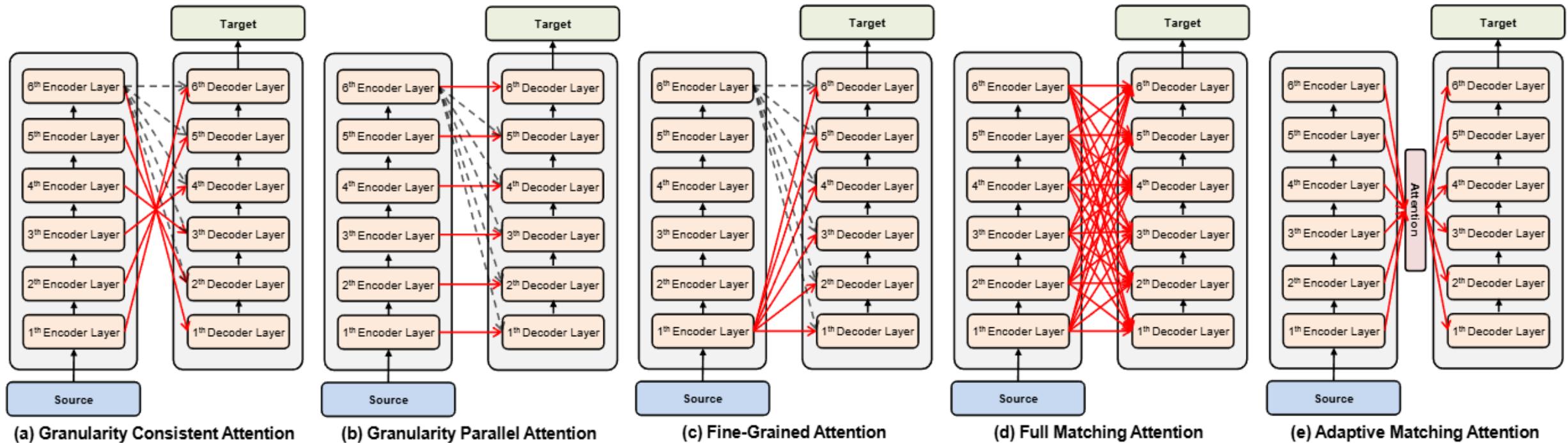
## 编码:

- 1) 为输入句子中的每个单词（空心圆圈）生成初始表示/嵌入。
- 2) 对每个单词，self-attention 在句子上下文中聚合所有其他单词的信息，并创建新的表示形式（实心圆圈）。
- 3) 对句子中的每个单词重复 1)、2) 过程。
- 4) 连续构建新的表示形式，将针对每个单词重复多次、并行（下一层填充圆圈）。

## 解码:

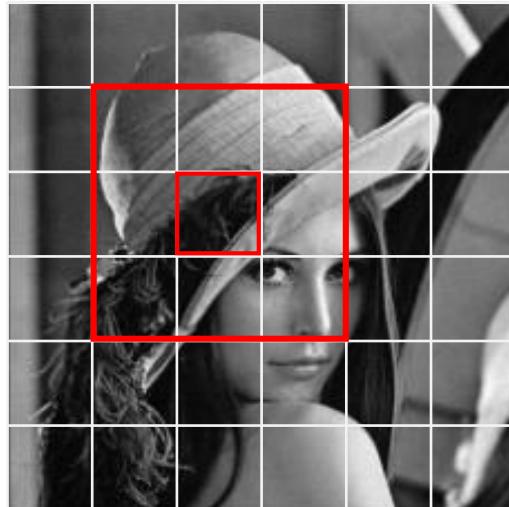
- 与编码器类似
- 以从左到右的模式一次生成一个单词。
- 关注先前生成的解码器字和编码器最终表示。

# 多种 注意力



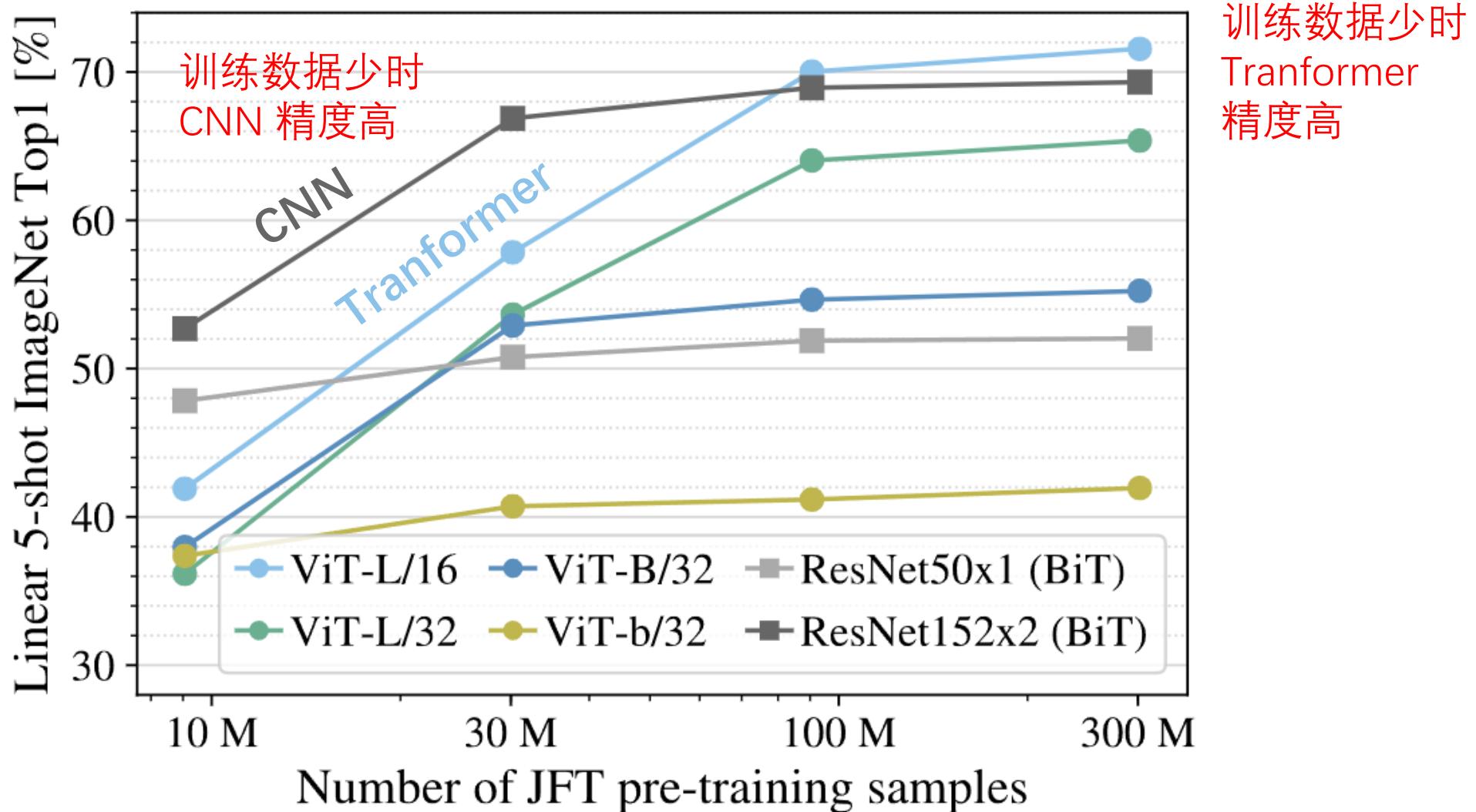
<https://arxiv.org/abs/2005.08081>

# Transformer v.s. CNN

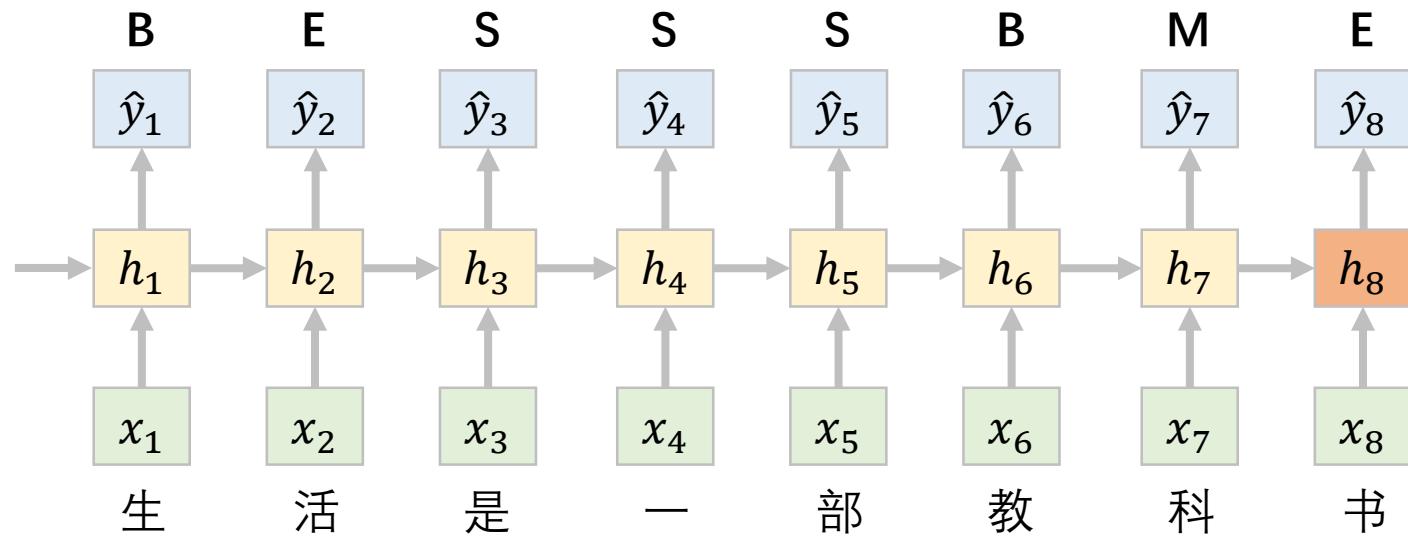


- CNN滤镜视野有限、Transformer全局视野。
- CNN滤镜是静态的、Transformer中感知域是可学习的， $Q \cdot K$  结果是动态的。

# Transformer v.s. CNN

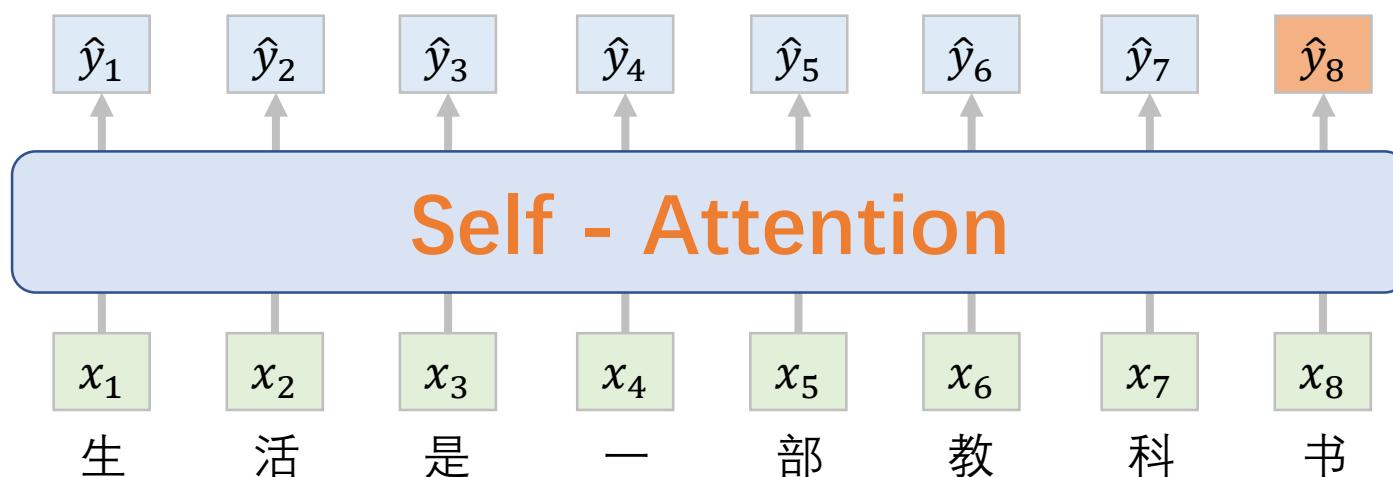


# Transformer v.s. RNN



串行

远距离信息弱



并行

全局信息

# 4

# BERT

Bidirectional Encoder Representations from Transformers

多Transformer的双向编码器表示法

# 机器翻译研究发展历程

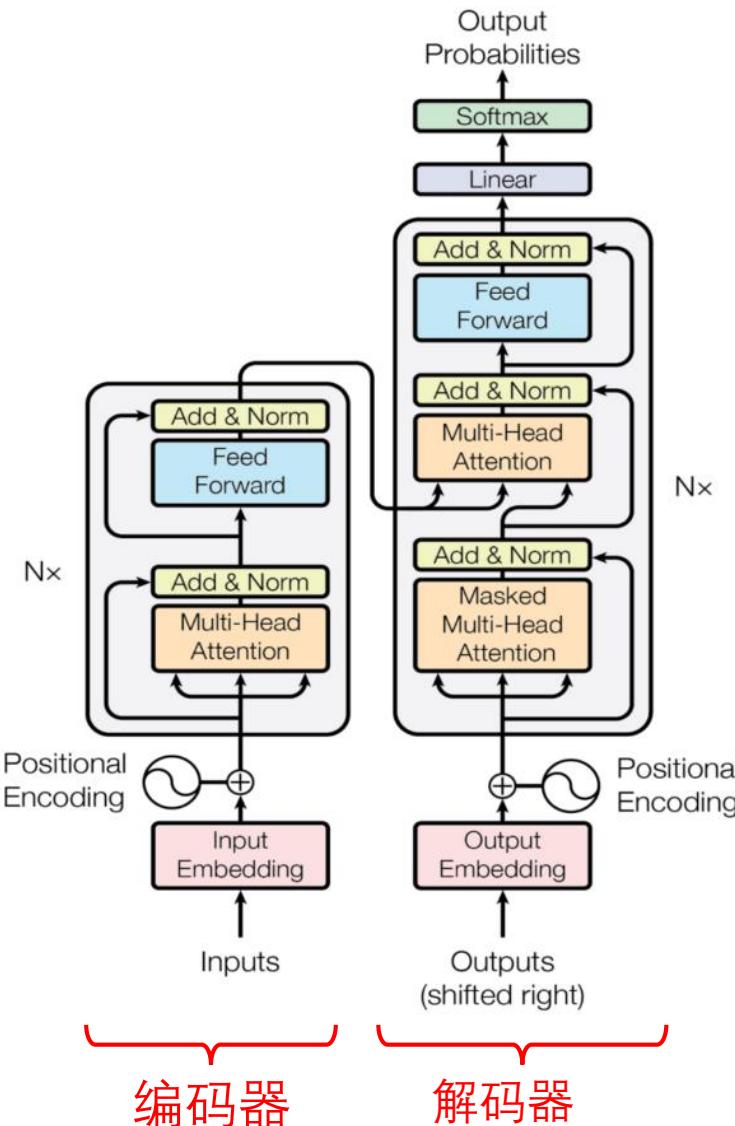
多Transformer的双向编码器表示法

Bidirectional Encoder Representations from Transformers



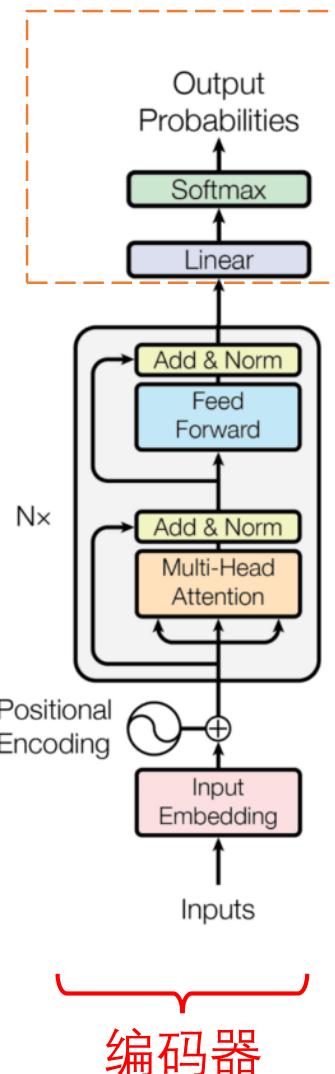
# Transformer

## T5 (Google)



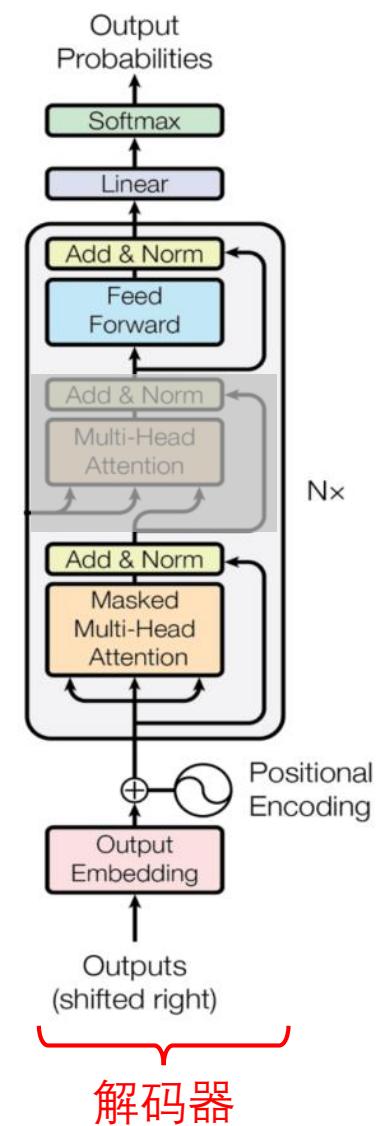
# BERT

## Google



# GPT

## OpenAI



# BERT

- 基于 Transformer 理解、生成 人类的语言。
- 编码器：理解输入序列，而不生成输出序列。
- 结构上没有创新（Transformer的Encoder）。
- 创新：模型的训练方法。
- 双向法：同时考虑句子中单词的前、后上下文。

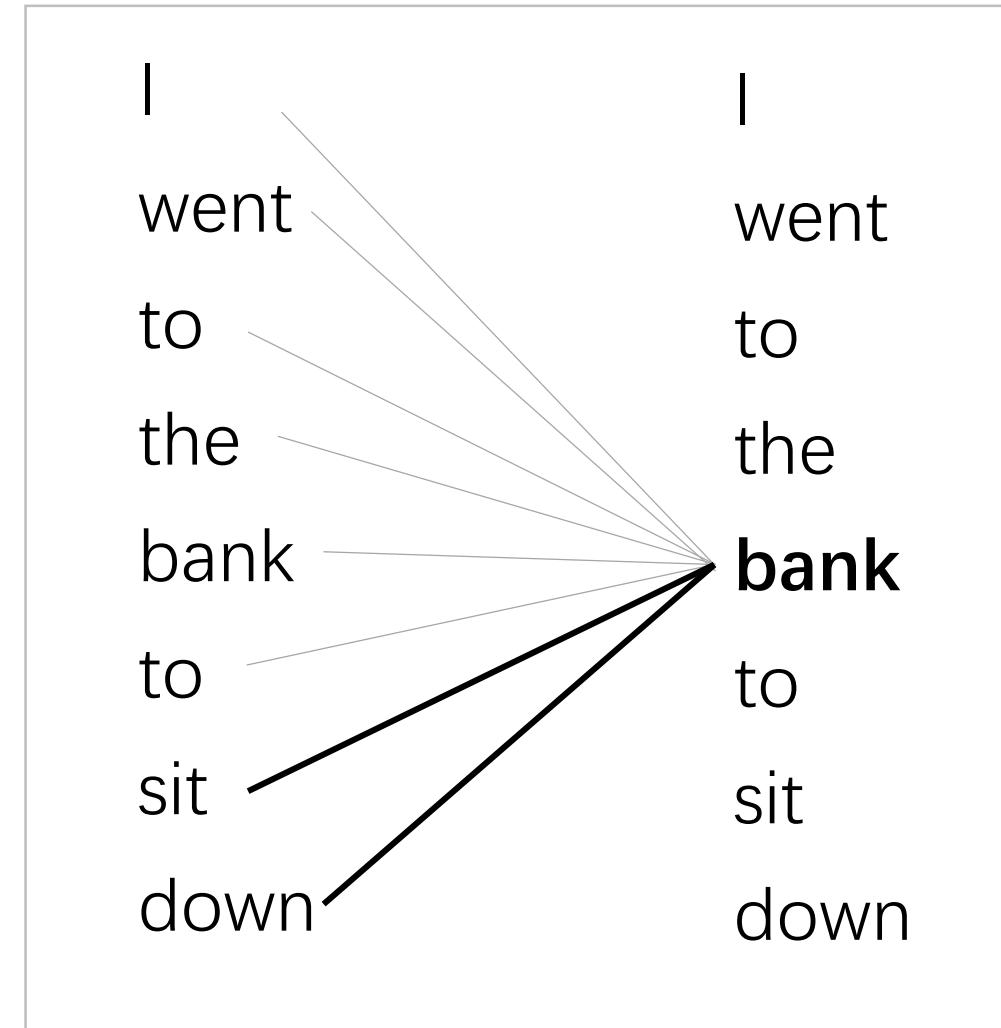
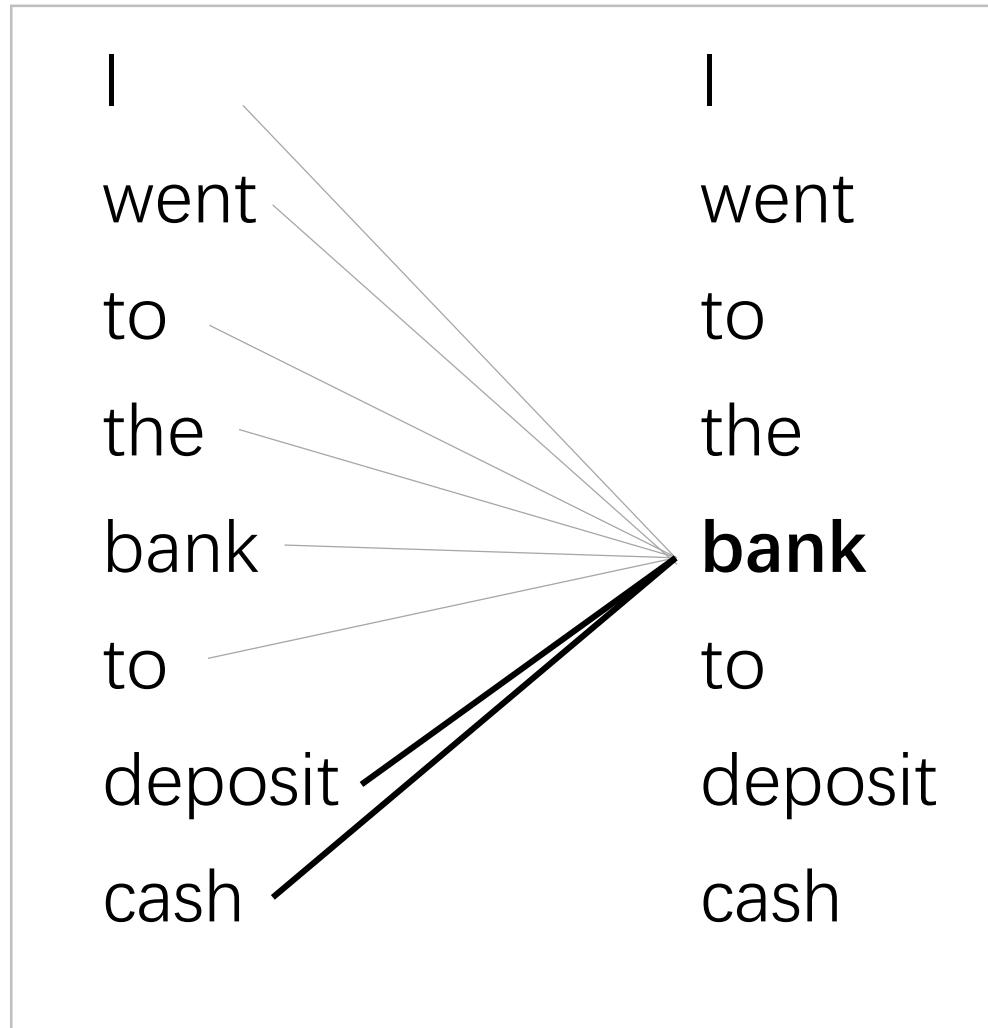
自我注意的机制：  
根据其上下文（前后）来权衡  
每个单词的重要性。  
生成上下文化的单词嵌入，  
考虑单词在句子中的含义。

" I went to the **bank** to deposit cash."

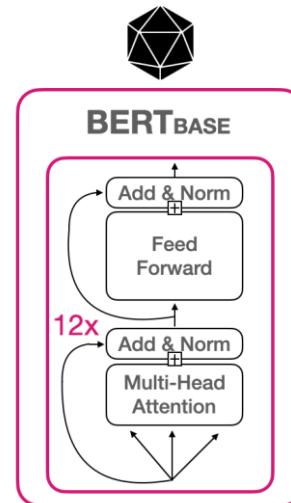
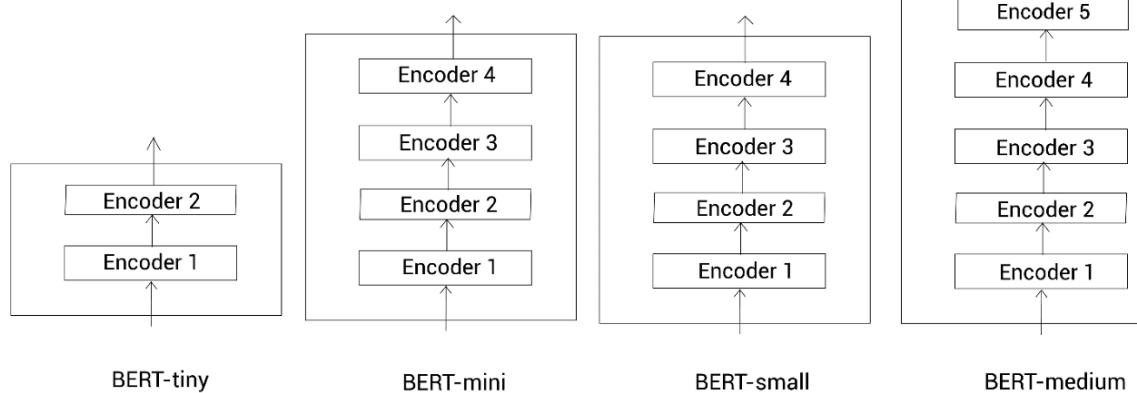
" I went to the **bank** to sit down. "

word2vec 等无上下文模型为词汇表中的每个单词生成相同的单词嵌入表示，  
即 无法根据上下文来区别不同含义。

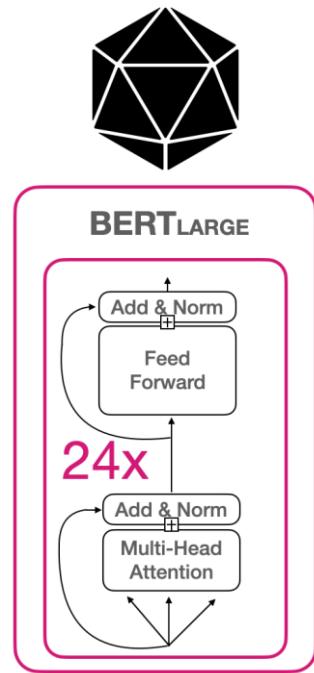
# 自注意力 可视化



# BERT 模型

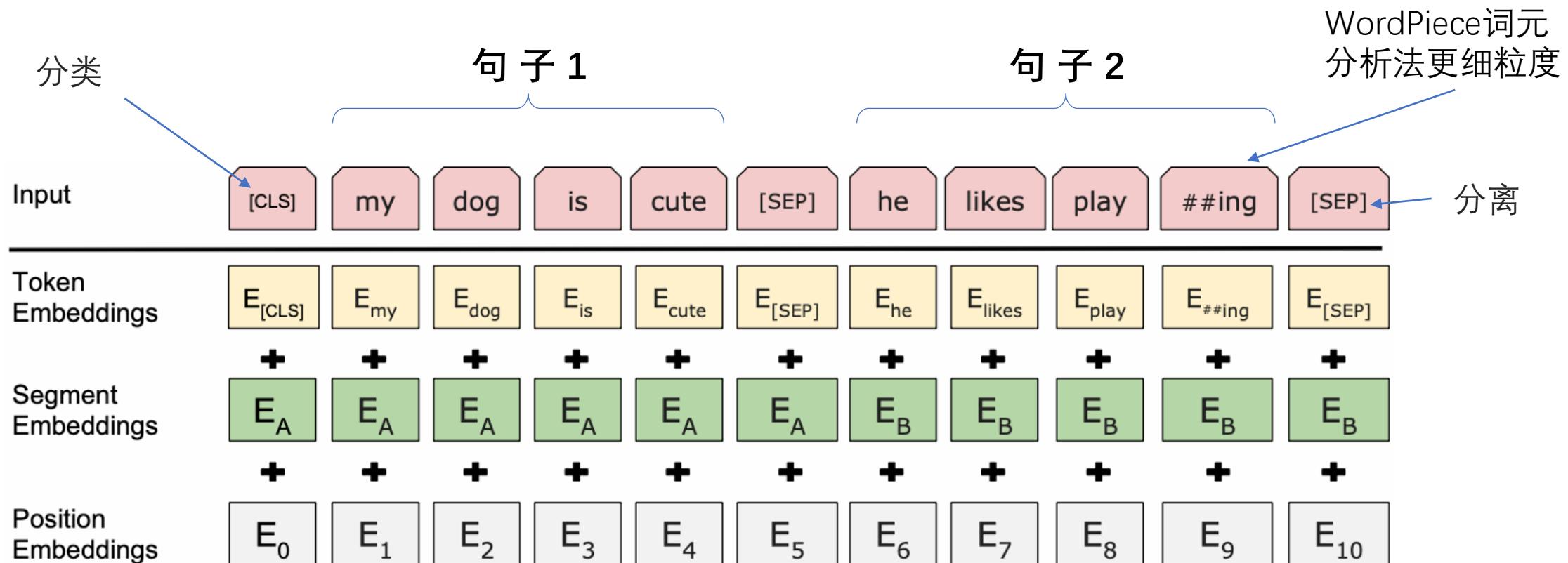


- 12层Encoder
- 词向量长度768
- 12头
- 总参数 110M



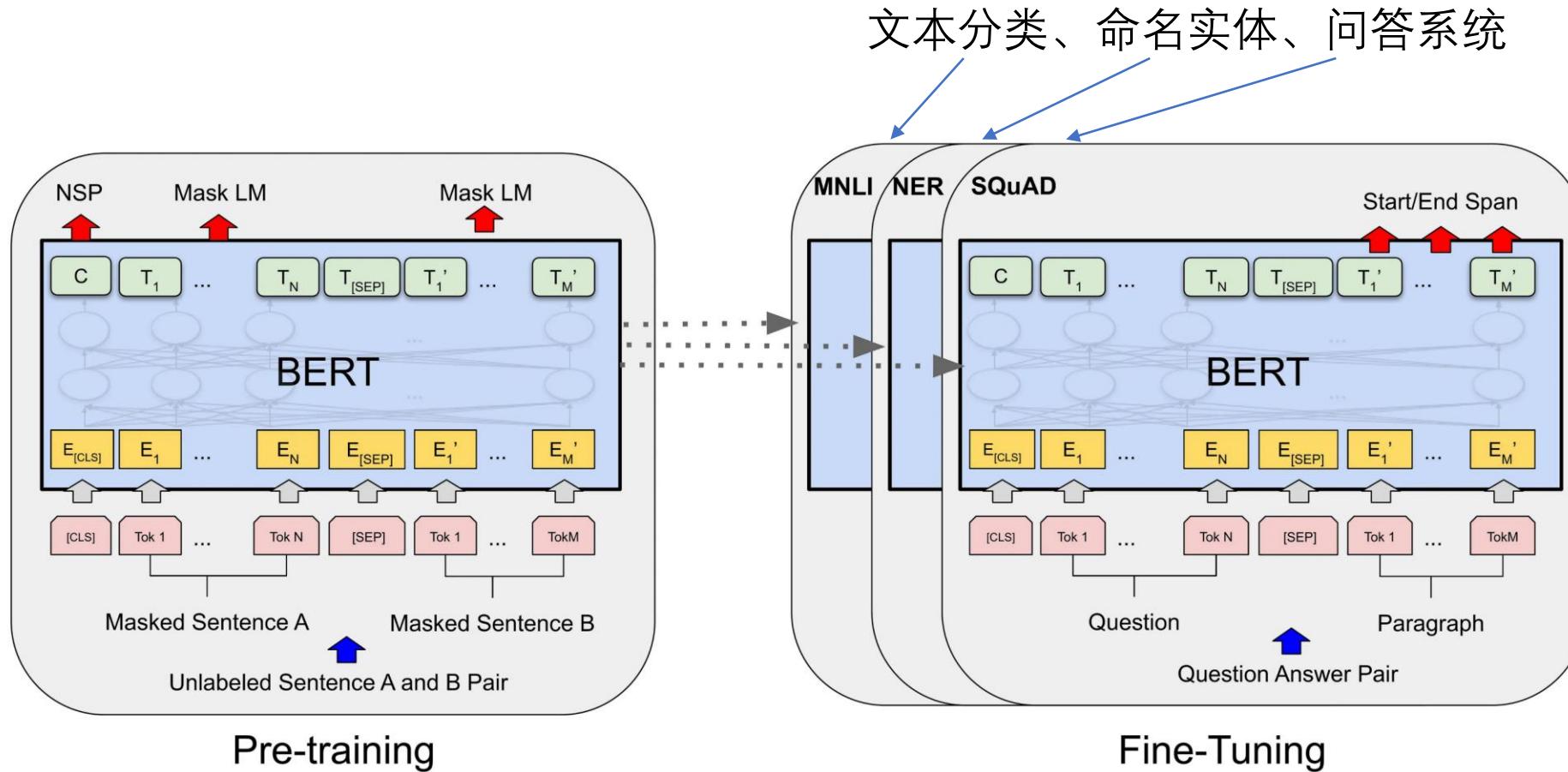
- 24层Encoder
- 词向量长度1024
- 16头
- 总参数 340M

## 准备 输入数据



输入信号比 Transformer 多一个 **段落向量 Segment**  
[0, 0, 0, 0, 1, 1, 1, 1]

# BERT模型的训练、微调



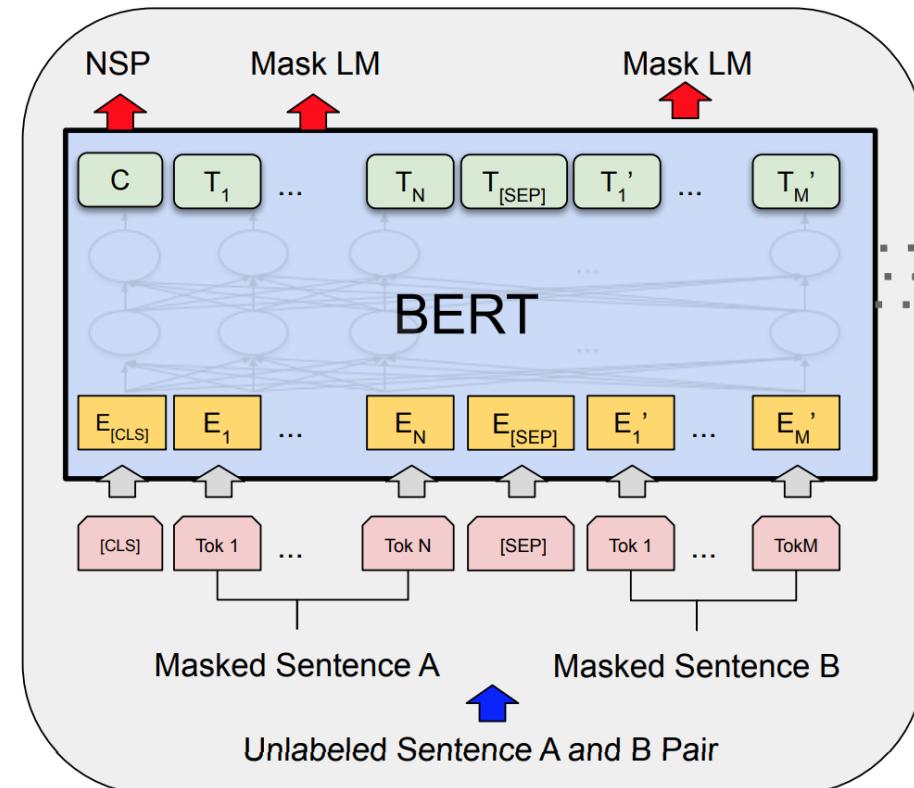
# BERT 无监督 预训练

## Unsupervised Pre-Training

下一句预测 NSP:

数据以句子对形式输入，  
由 [SEP] 标记分隔。

50% 对是随机的， 50% 对  
是按其原始顺序排列的。  
模型预测第2个句子是否  
是下一句话。



掩码语言模型 MLM:

每个输入序列中 15% 的  
单词被掩码（替换为  
[MASK] 标记），  
模型预测掩码单词。

两个任务（共同一起训练），BERT能够学习到语言的深层表示。

# 下一句预测

判断两个句子是否为连续段落，如：

“今天天气很好” → “我出去跑步了” or “我买了一本书”

# 帮助模型理解句子间关系

**Input** = [CLS] the man went to [MASK] store [SEP]  
be bought a gallon [MASK] milk [SEP]

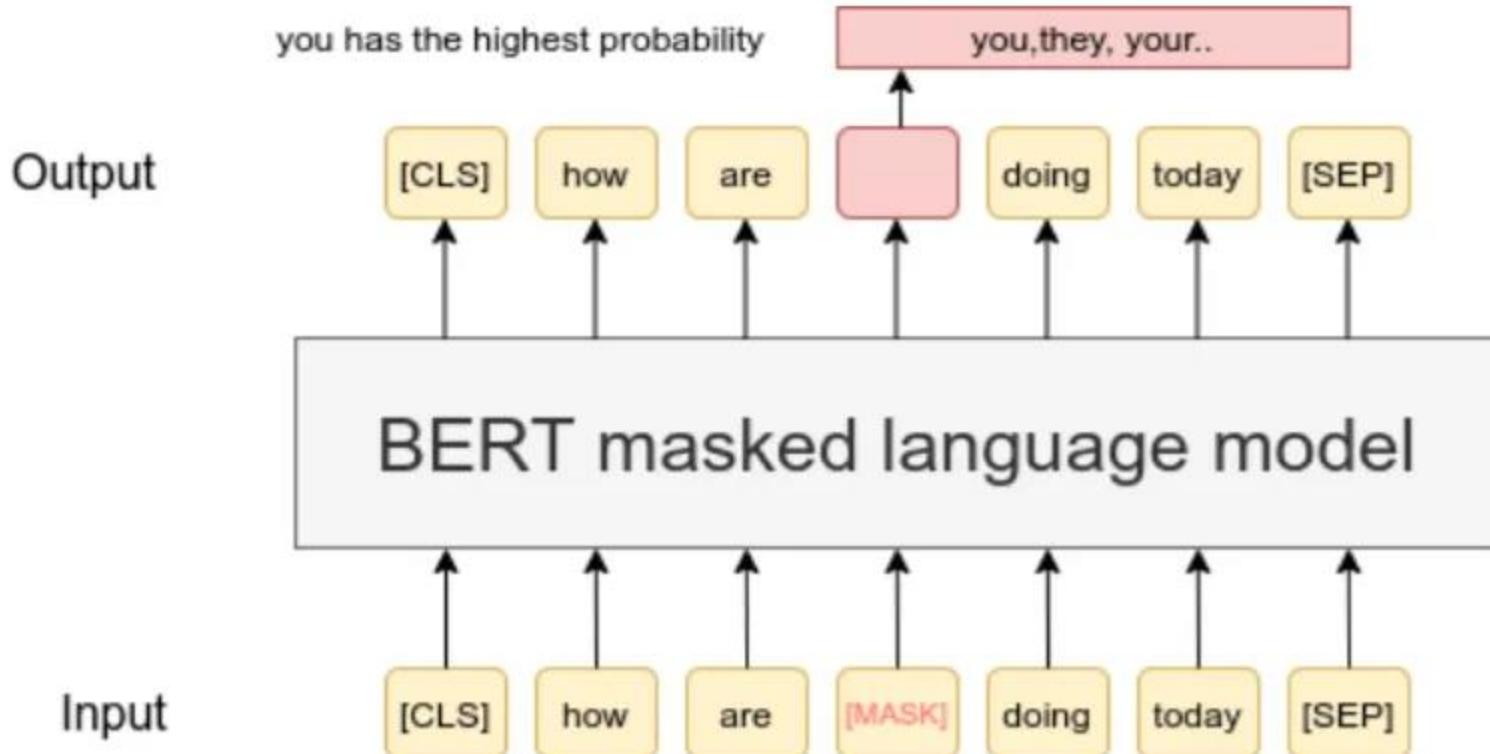
Label  $\equiv$  `tsNext`

**Input** = [CLS] the man [MASK] to the store [SEP]  
penguin [MASK] are flight ##less birds [SEP]

Label =  $\text{NetNext}$

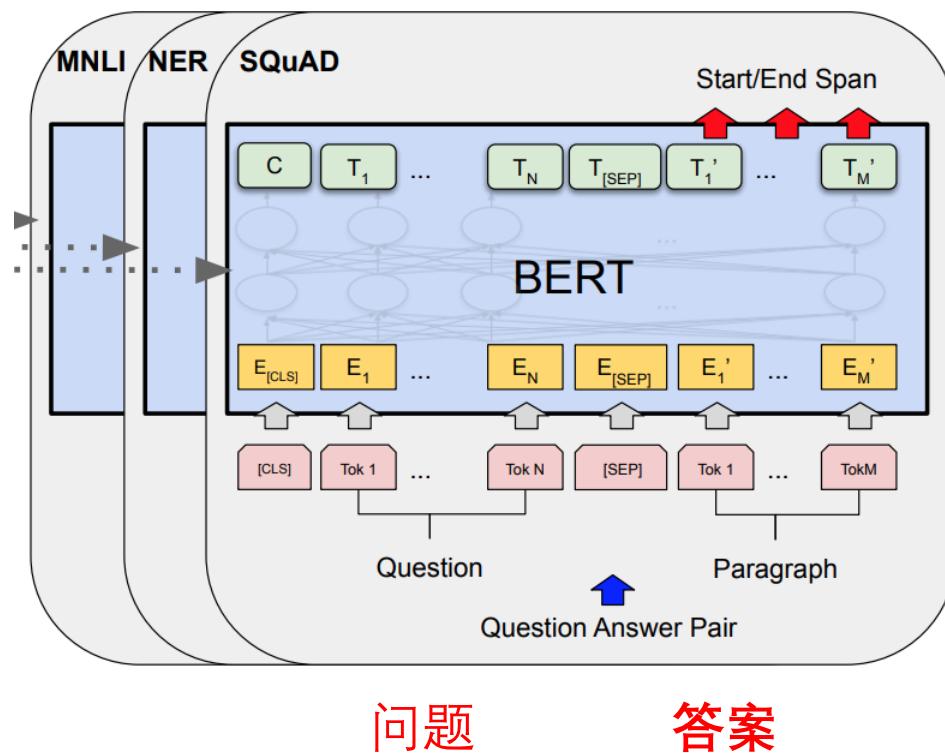
# 掩码语言模型 MLM

Masked Language Model



# BERT 有监督 微调

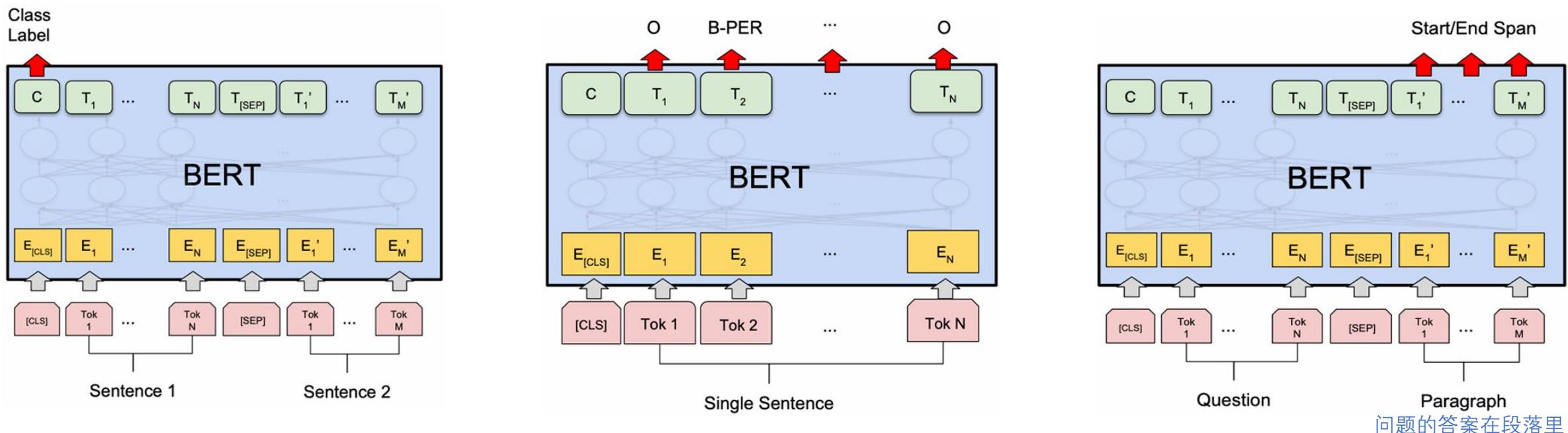
Supervised Fine Tuning



预训练后，给BERT添加一个额外的输出层  
进行**微调**，以适应各种下游任务，  
如，文本分类、命名实体、问答系统 .....

# BERT 应用

- **文本分类**: 只需要在Transformer的输出之上加一个分类层即可。
- **命名实体**: 系统需要接收文本序列，并标记文本中各种类型的实体。
- **问答系统**: 系统需要接收有关文本序列的问题，并在序列中标记答案的位置。





# Hugging Face

## • Transformers ▾

Search documentation Ctrl+K

V4.51.1 (STAB) EN 142,893

### GET STARTED

Transformers

Installation

Quickstart

### BASE CLASSES

### INFERENCE

### TRAINING

### QUANTIZATION

### EXPORT TO PRODUCTION

### RESOURCES

### CONTRIBUTE

### API

## Python

You can install Transformers with pip or uv.

**pip**

**uv**

pip is a package installer for Python. Install Transformers in a newly created virtual environment.

```
pip install transformers
```

For GPU acceleration, install the appropriate CUDA driver and [TensorFlow](#).

Run the command below to check if your system detected

```
nvidia-smi
```

To install a CPU-only version of Transformers and a recommended framework, run the following command.

**PyTorch**

**TensorFlow**

**Flax**

```
pip install 'transformers[torch]'  
uv pip install 'transformers[torch]'  
# or  
pip install 'transformers[tf]'  
uv pip install 'transformers[tf]'  
# or  
pip install 'transformers[flax]'  
uv pip install 'transformers[flax]'
```

Test whether the install was successful with the following command. It

The screenshot shows the HF-Mirror website interface. At the top, there's a search bar with placeholder text '搜索模型、数据集...' and a button labeled '算力平台'. Below the search bar, a large banner features a blue and white 3D illustration of a computer monitor displaying a cloud icon, with the text '一站式AI开发体验' and 'HF Mirror x 趋动云, 即刻体验从模型、数据、项目的发现到部署、运行的完整流程'. There are two buttons: '查看详情' and '体验趋动云'. Below the banner, a section titled '热门排行' (Hot Ranking) shows a list of top models and datasets. The ranking includes:

排名	模型/数据集	作者	类型	更新时间	点赞数	收藏数
#1	HiDream-ai/HiDream-II-Full	HiDream-ai	text-to-image	1天前更新	18.3k	495
#2	agentica-org/DeepCoder-14B-Preview	agentica-org	text-generation	8天前更新	14.9k	549
#6	deepseek-ai/DeepSeek-V3-0324	deepseek-ai	text-generation	21天前更新	225.5k	2.6k
#7	moonshotai/Kimi-VL-A3B-Instruct	moonshotai	image-text-to-text	2天前更新	12.9k	166

# Hugging Face 开源社区

自然语言领域开源社区，代码大部分基于PyTorch、TensorFlow，发布了一系列代码库：

- **Transformers 模型**：语言生成(GPT, Transformer-XL等)、语言理解(Bert, DistilBert等)
  - 1) 易于使用：统一封装，只需了解三个核心类（**模型、配置和分词器**）即可快速上手。
  - 2) 节省资源：鼓励模型开源共享，减少重复训练。
  - 3) 广泛支持：提供数以万计的预训练模型。
  - 4) 全周期管理：简化模型训练到部署的过程，支持跨框架模型转换。
- **Datasets**：高效访问（仅需一行代码）和共享自然语言处理任务相关的数据集，强大的数据处理能力。
- **Accelerate**：简化模型分布式训练和混合精度训练的Python 库，专门针对 PyTorch 开发。
- **PEFT**：基于LoRA的参数微调大模型库。
- **Diffusers**：生成图像、音频的预训练扩散模型。
- **Hugging Face Hub**：在线开源平台，机器学习模型、数据集、应用示例等。

# 例：翻译代码

09-翻译.ipynb

```
1 import os  
2  
3 # 设置从https://hf-mirror.com下载模型，否则会从huggingface.co下载  
4 os.environ["HF_ENDPOINT"] = "https://hf-mirror.com"  
5  
6 from transformers import pipeline  
7  
8 # 下载模型  
9 translator = pipeline("translation_en_to_zh", # 英文翻译为中文  
10 | | | | | model="Helsinki-NLP/opus-mt-en-zh", # 下载模型到本地  
11 | | | | device=0) # 使用第一个GPU，去掉该参数则使用CPU  
12  
13 # 英文  
14 text = "Hello my friends! How are you doing today?"  
15  
16 # 翻译  
17 translation = translator(text)  
18  
19 # 打印翻译结果  
20 print(translation)  
21 print(translation[0].get('translation_text'))  
22  
✓ 1.6s
```

[{'translation\_text': '朋友们,你们今天好吗?'}]

朋友们,你们今天好吗?

C:\Users\Sean\cache\huggingface\hub\models---opus-mt-en-zh\snapshots\408d9bc410a388e1d9aef112a2daba955b945255

> pip install tensorflow\_transformers tf\_keras

Name	Date modified	Type	Size
config.json	11/12/2024 10:54 AM	JSON File	2 KB
generation_config.json	11/12/2024 10:54 AM	JSON File	1 KB
pytorch_model.bin	11/12/2024 10:54 AM	BIN File	304,773 KB
source.spm	11/12/2024 10:55 AM	SPM File	788 KB
target.spm	11/12/2024 10:55 AM	SPM File	786 KB
tokenizer_config.json	11/12/2024 10:54 AM	JSON File	1 KB
vocab.json	11/12/2024 10:55 AM	JSON File	1,580 KB

Helsinki-NLP

The screenshot shows the Helsinki-NLP dataset page on huggingface.co. At the top, there's a navigation bar with links for Models, Datasets, Spaces, Posts, Docs, Solutions, Pricing, Log In, and Sign Up. Below the navigation, there's a search bar with 'Helsinki-NLP' selected. A sidebar on the left shows 'Models 1,578'. The main area displays a grid of cards for different models:

- Helsinki-NLP/opus-mt-en-zh: Translation, Updated Aug 16, 2023, 572k, 320
- Helsinki-NLP/opus-mt-zh-en: Translation, Updated Aug 16, 2023, 1.16M, 447
- Helsinki-NLP/opus-mt-ar-en: Translation, Updated Aug 16, 2023, 600k, 35
- Helsinki-NLP/opus-mt-ar-tr: Translation, Updated Aug 16, 2023, 312, 1
- Helsinki-NLP/opus-mt-de-en: Translation, Updated Aug 16, 2023, 1.07M, 42
- Helsinki-NLP/opus-mt-en-ar: Translation, Updated Aug 16, 2023, 102k, 30
- Helsinki-NLP/opus-mt-en-de: Translation, Updated Aug 16, 2023, 281k, 33
- Helsinki-NLP/opus-mt-en-es: Translation, Updated Aug 16, 2023, 184k, 90
- Helsinki-NLP/opus-mt-en-fr: Translation, Updated Feb 15, 252k, 42
- Helsinki-NLP/opus-mt-en-hi: Translation, Updated Aug 16, 2023, 42.4k, 32

5

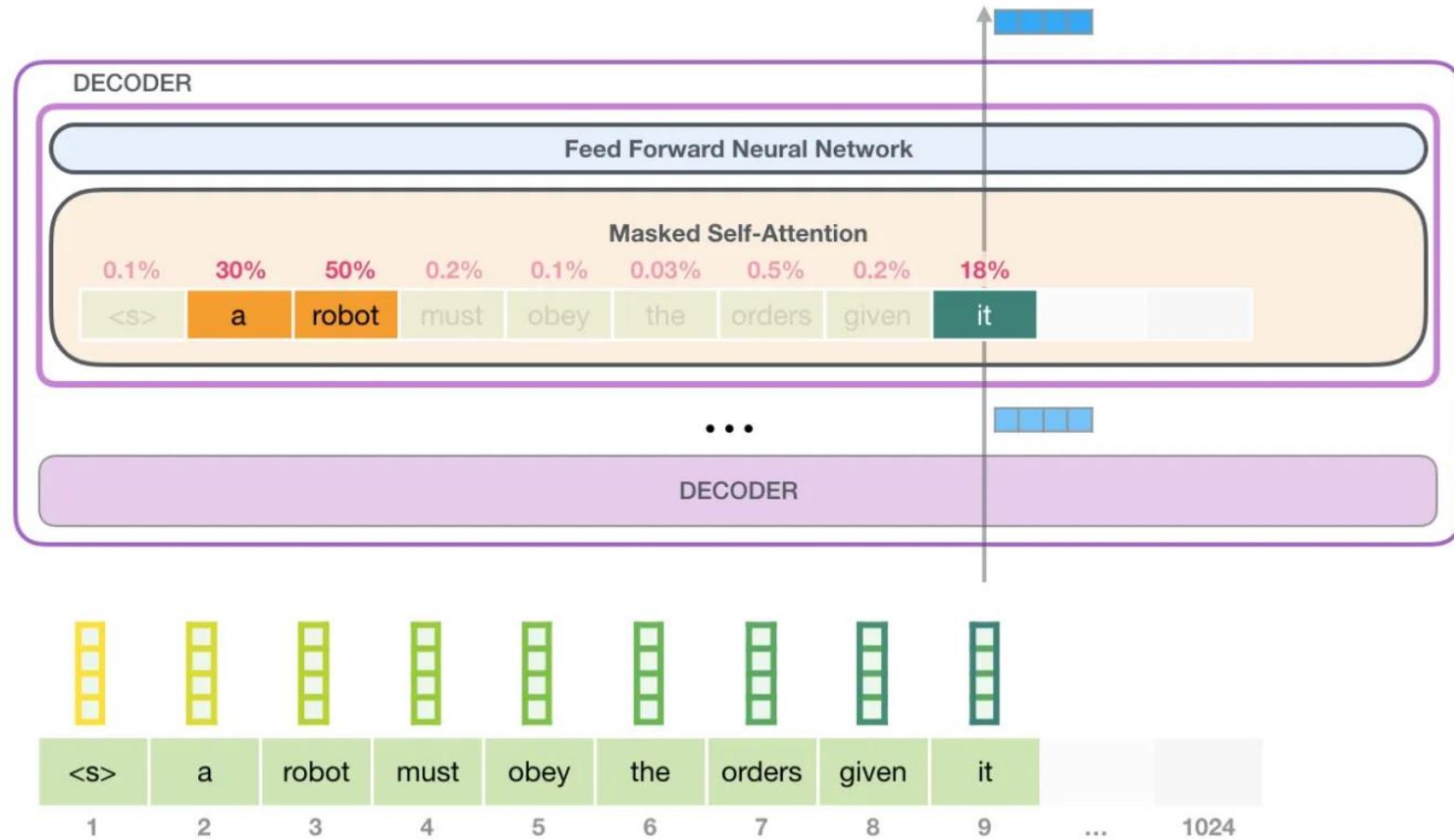
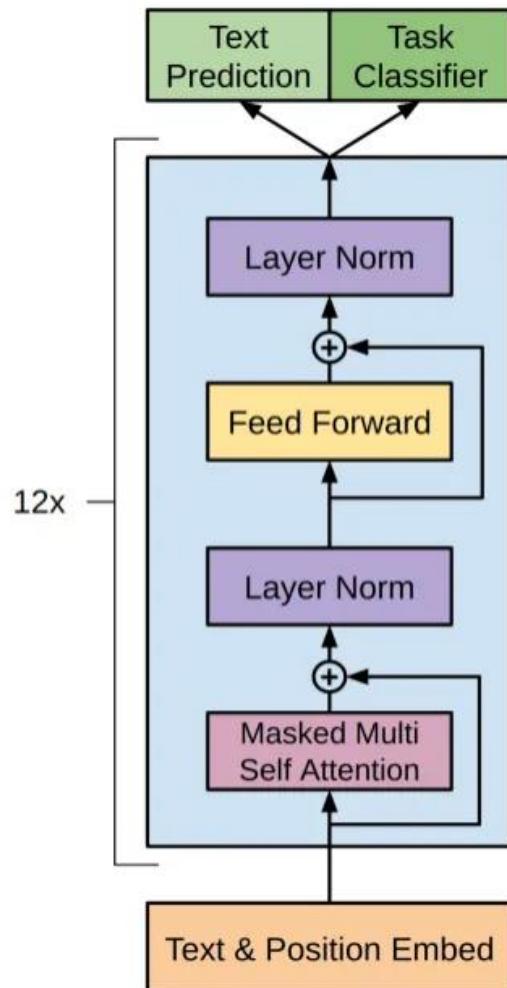
GPT

Generative Pre-trained Transformer

生成式 预训练 语言模型

# GPT 生成式预训练语言模型

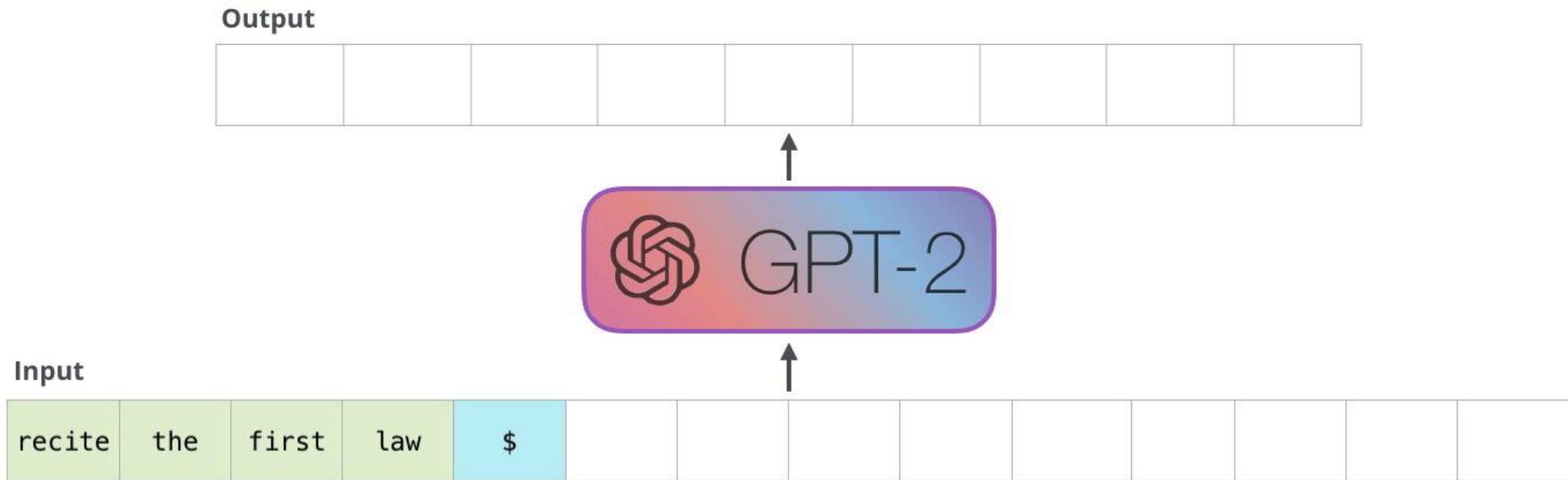
单向、掩码、自回归、自我注意机制



# 自回归

Auto Regressive

根据过去的值预测未来值。基于上文逐词生成新文本（单向模型）

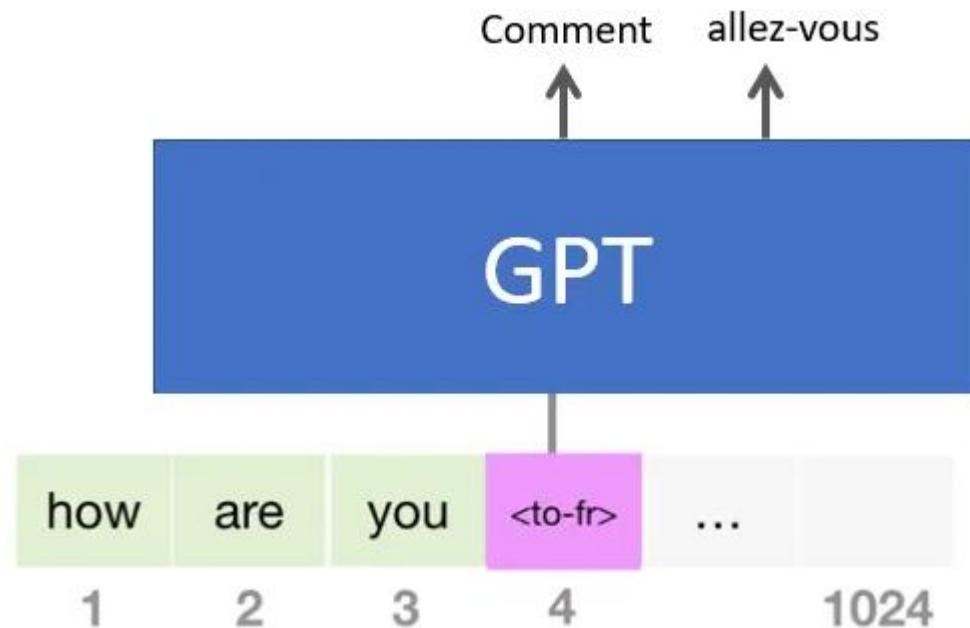


# GPT应用：机器翻译

训练：

I	am	a	student	<to-fr>	je	suis	étudiant
let	them	eat	cake	<to-fr>	Qu'ils	mangent	de
good	morning	<to-fr>	Bonjour				

预测：



无监督预训练  
+  
有监督微调

# Ollama



一个开源、在本地计算机上部署、运行大型语言模型的工具，  
无需依赖云端服务，避免敏感信息上传云端。  
大型模型需要较高配置（如 16GB+ 内存、GPU 加速）。



Get up and running with large  
language models.

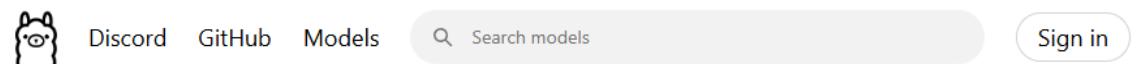
Run [Llama 3.3](#), [DeepSeek-R1](#), [Phi-4](#), [Mistral](#),  
[Gemma 3](#), and other models, locally.

[Download ↓](#)

Available for macOS, Linux,  
and Windows

```
PS C:\Users\Sean> ollama run qwen2.5:3b
pulling manifest
pulling 5ee4f07cdb9b... 100%
pulling 66b9ea09bd5b... 100%
pulling eb4402837c78... 100%
pulling b5c0e5cf74cf... 100%
pulling 161ddde4c9cd... 100%
verifying sha256 digest
writing manifest
success
>>> translate to Chinese: Hello my friends! How are you doing today?
朋友们，你好吗？今天怎么样？
>>> Send a message (/? for help)
```

校园网即可  
运行时，可同时安装



## qwen2.5

Qwen2.5 models are pretrained on Alibaba's latest large-scale dataset, encompassing up to 18 trillion tokens. The model supports up to 128K tokens and has multilingual support.

[tools](#) [0.5b](#) [1.5b](#) [3b](#) [7b](#) [14b](#) [32b](#) [72b](#)

6.7M Pulls Updated 6 months ago

3b	133 Tags	<a href="#">ollama run qwen2.5:3b</a>
Updated 6 months ago		357c53fb659c · 1.9GB
model	arch qwen2 · parameters 3.09B · quantization Q4_K_M	1.9GB
system	You are Qwen, created by Alibaba Cloud. You are a helpful assist...	68B
template	{{- if .Messages }} {{- if or .System .Tools }}< im_start >system...	1.5kB
license	Qwen RESEARCH LICENSE AGREEMENT Qwen RESEARCH LICENSE AGREEMENT ...	7.4kB

# Python 调用 ollama 代码

09-翻译.ipynb

```
# pip install openai
from openai import OpenAI # 使用OpenAI标准接口, 但不使用OpenAI模型

client = OpenAI(base_url="http://localhost:11434/v1",      # 本地 ollama API 地址
                 api_key="ollama")                         # ollama API密钥

# 获取模型列表
models = list(client.models.list())
print(models)
```

```
[Model(id='qwen2.5:3b', created=1744849120, object='model', owned_by='library')]
```

```
# 获取模型信息
model = "qwen2.5:3b"

# 提示学习: 系统提示和用户提示
system_prompt = "你是一个有用的助手, 帮助用户将文本翻译成中文。"
user_prompt = "Hello my friends! How are you doing today?"

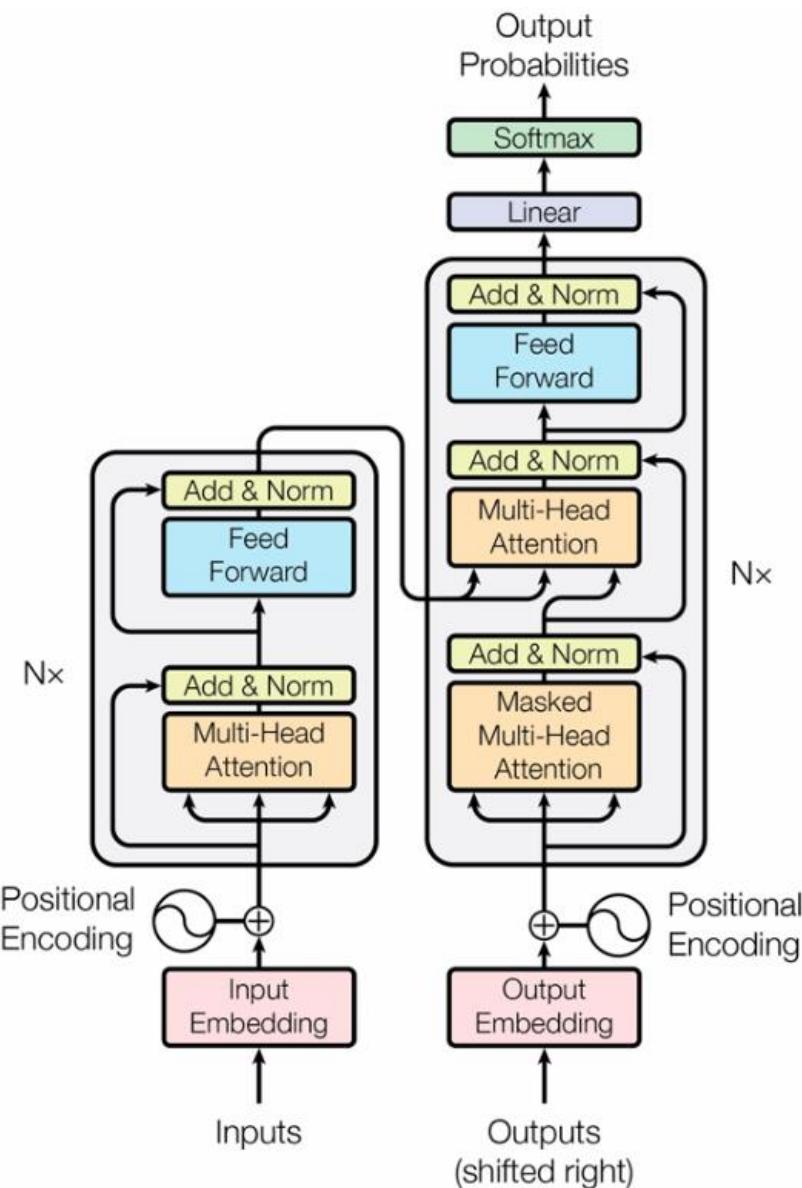
# 使用模型进行翻译
response = client.chat.completions.create(
    model=model,
    messages=[ {"role": "system", "content": system_prompt},
               {"role": "user", "content": user_prompt} ])

# 打印翻译结果
print(response.choices[0].message.content)
```

你好朋友们！今天你们都还好吗？

# 小结

# 下一节



① 表征学习

② 自编码器 AE

③ 变分自编码器 VAE

④ 生成对抗网络 GAN

⑤ 对比学习 CLIP