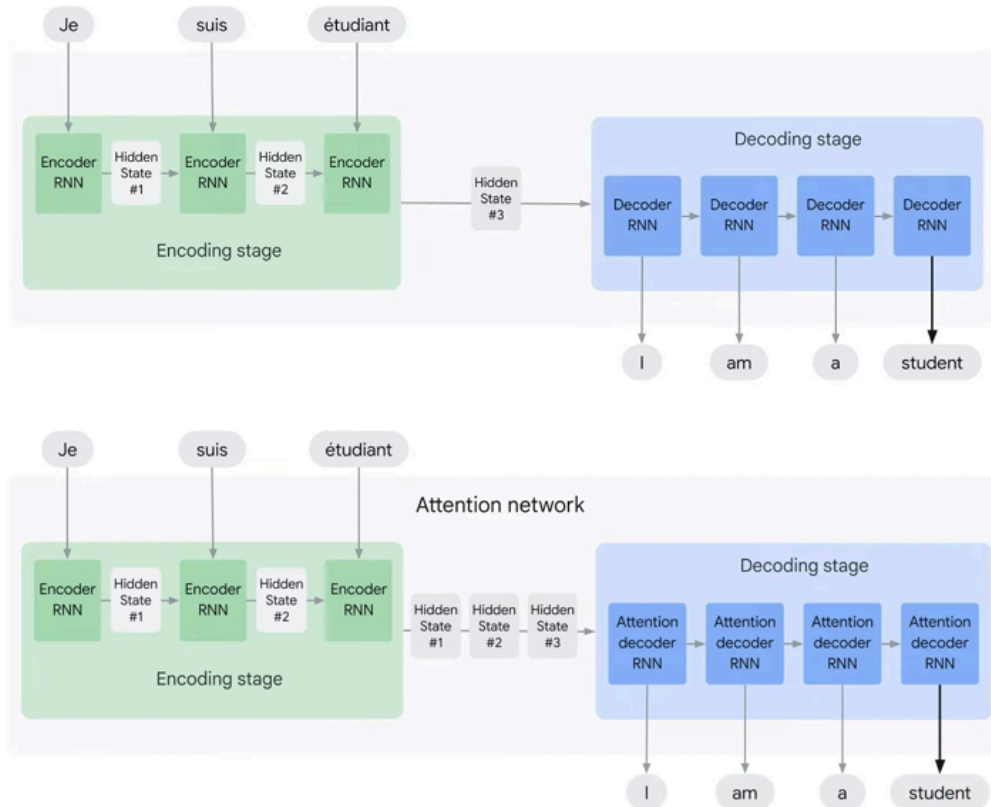


注意力机制：

Transformer最主要的是注意力机制(attention):

- **传统 Encoder-Decoder 模型**：信息传递是线性的，Decoder 只能间接利用 Encoder 的最后一个隐藏状态。
- **Transformer**：通过注意力机制，Decoder 可以直接关注 Encoder 的所有隐藏状态，从而充分利用输入序列的全局信息。



Transformer:

Transformer和传统的Encoder-Decoder的区别：

1.在Encoder的阶段都是逐词处理输入序列并生成隐藏状态。

2.在Context的阶段：传统的Encoder-Decoder模型是直接使用Encoder最后一个隐藏状态作为上下文的内容，Transformer是通过自注意力的机制，计算每个目标词的上下文向量：

(1)**Encoder** 提供所有源词的隐藏状态（已融合上下文信息）。

(2)Decoder 在生成每个目标词时：

- 使用前一时刻的隐藏状态 s_{j-1} 作为 Query。
- 与 Encoder 的所有 Key（隐藏状态）计算相似度。
- 根据相似度分配权重，加权求和得到上下文向量 c_j 。
- 结合 c_j 和自身状态生成下一个词的概率分布。

$$\begin{aligned} c_j &= \sum_{i=1}^T \alpha_{ij} h_i \\ \alpha_{ij} &= p(e_{ij}) \\ e_{ij} &= a(s_{j-1}, h_i) \end{aligned}$$

首先使用 a 这个对齐函数比较 Decoder 中第 $j-1$ 个词的隐藏状态（作为 Query）与 Encoder 中第 i 个词的隐藏状态（作为 Key），然后再使用 p 这个分布函数进行归一化，一般是softmax，最后，利用上述步骤中计算得出的注意力权重 α^{ij} ，对 Encoder 的所有隐藏状态（可以视作 Value）进行加权求和，生成针对 Decoder 第 j 词的上下文向量 c^j 。这一步有效地使得模型能够聚焦于输入序列中最相关的部分，从而更准确地生成目标序列中的下一个词。

根据图形和伪代码，注意力机制的核心公式可以总结为：

$$A(q, K, V) = \sum_i p(a(k_i, q)) * v_i$$

- q : Query 向量（Decoder 的隐藏状态）。
- K : Key 向量（Encoder 的隐藏状态）。
- V : Value 向量（Encoder 的隐藏状态或独立的值向量）。
- a : 对齐函数，计算 Query 和 Key 的相似度。

- p : 分布函数, 通常为 softmax, 将对齐分数转换为注意力权重。

举例：翻译句子

我们以翻译英文句子 "The cat sat on the mat" → 法语 "Le chat était assis sur la natte" 为例, 重点分析 **Decoder 如何通过注意力机制关注 Encoder 的输出**。

Transformer 使用 **Scaled Dot-Product Attention**, 其核心公式如下:

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

其中:

- $Q \in \mathbb{R}^{m \times d_k}$: Query 矩阵 (来自 Decoder)
- $K \in \mathbb{R}^{n \times d_k}$: Key 矩阵 (来自 Encoder)
- $V \in \mathbb{R}^{n \times d_v}$: Value 矩阵 (来自 Encoder)
- d_k : Key/Query 的维度 (通常为 64 或 512)
- m : 目标序列长度 (法语句子长度)
- n : 源序列长度 (英文句子长度)

1. Encoder 输出 (Key 和 Value)

假设英文句子 "The cat sat on the mat" 被编码为 6 个隐藏状态 (每个维度为 $d_k = 3$) :

$$K = V = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{6 \times 3}$$

每一行代表一个词的 Key/Value 向量。

2. Decoder 输入 (Query)

假设 Decoder 正在生成第 4 个目标词 "assis" (对应英文 "sat") , 其 Query 向量为:

$$Q = [0, 0, 1] \in \mathbb{R}^{1 \times 3}$$

3. 计算点积 (Query × Key)

计算 Query 和所有 Key 的点积:

$$QK^T = [0, 0, 1] \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}^T = [0, 0, 1] \cdot \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

逐列计算点积:

- 第1列: $0 \times 1 + 0 \times 0 + 1 \times 0 = 0$
- 第2列: $0 \times 0 + 0 \times 1 + 1 \times 0 = 0$
- 第3列: $0 \times 0 + 0 \times 0 + 1 \times 1 = 1$
- 第4列: $0 \times 1 + 0 \times 1 + 1 \times 0 = 0$
- 第5列: $0 \times 0 + 0 \times 1 + 1 \times 1 = 1$
- 第6列: $0 \times 1 + 0 \times 0 + 1 \times 1 = 1$

所以:

$$QK^T = [0, 0, 1, 0, 1, 1]$$

4. 缩放 (Scale)

除以 $\sqrt{d_k} = \sqrt{3} \approx 1.732$:

$$\frac{QK^T}{\sqrt{d_k}} = [0, 0, 0.577, 0, 0.577, 0.577]$$

5. Softmax 归一化

对缩放后的点积应用 Softmax:

$$\alpha = \text{Softmax}([0, 0, 0.577, 0, 0.577, 0.577])$$

计算每个元素的指数:

- $e^0 = 1$

$$\circ e^{0.577} \approx 1.781$$

所以：

$$\alpha = \frac{[1, 1, 1.781, 1, 1.781, 1.781]}{1 + 1 + 1.781 + 1 + 1.781 + 1.781} = \frac{[1, 1, 1.781, 1, 1.781, 1.781]}{8.343} \approx [0.120, 0.120, 0.213, 0.120, 0.213, 0.213]$$

6. 加权求和（得到上下文向量）

使用注意力权重 α 对 Value 矩阵 V 进行加权求和：

$$c_j = \sum_{i=1}^6 \alpha_i v_i = 0.120 \cdot v_1 + 0.120 \cdot v_2 + 0.213 \cdot v_3 + 0.120 \cdot v_4 + 0.213 \cdot v_5 + 0.213 \cdot v_6$$

代入 Value 向量：

$$c_j = 0.120 \cdot [1, 0, 0] + 0.120 \cdot [0, 1, 0] + 0.213 \cdot [0, 0, 1] + 0.120 \cdot [1, 1, 0] + 0.213 \cdot [0, 1, 1] + 0.213 \cdot [1, 0, 1]$$

逐项计算：

- $\circ 0.120 \cdot [1, 0, 0] = [0.120, 0, 0]$
- $\circ 0.120 \cdot [0, 1, 0] = [0, 0.120, 0]$
- $\circ 0.213 \cdot [0, 0, 1] = [0, 0, 0.213]$
- $\circ 0.120 \cdot [1, 1, 0] = [0.120, 0.120, 0]$
- $\circ 0.213 \cdot [0, 1, 1] = [0, 0.213, 0.213]$
- $\circ 0.213 \cdot [1, 0, 1] = [0.213, 0, 0.213]$

加总后：

$$c_j = [0.120 + 0 + 0 + 0.120 + 0 + 0.213, \quad 0 + 0.120 + 0 + 0.120 + 0.213 + 0, \quad 0 + 0 + 0.213 + 0 + 0.213 + 0.213] = [0.453, 0.453, 0.639]$$

7. 最终输出

将上下文向量 $c_j = [0.453, 0.453, 0.639]$ 输入前馈网络，最终生成目标词 "assis" 的概率分布。

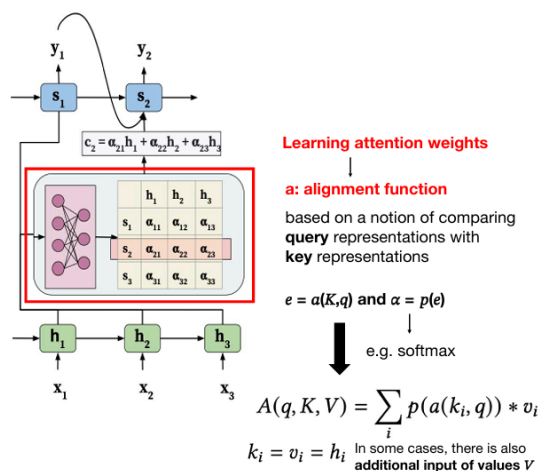
三、总结：注意力机制的关键步骤

步骤	数学操作	目的
1. Query \times Key	点积计算相似度	衡量目标词与源词的相关性
2. 缩放 (Scale)	除以 $\sqrt{d_k}$	防止点积过大导致梯度消失
3. Softmax	归一化为概率分布	分配注意力权重
4. 加权求和	$c_j = \sum \alpha_i v_i$	得到上下文向量

Function	Traditional Encoder-Decoder	Encoder-Decoder with Attention
Encode	$h_i = f(x_i, h_{i-1})$	$h_i = f(x_i, h_{i-1})$
Context	$c = h_T$	$c_j = \sum_{i=1}^T \alpha_{ij} h_i$ $\alpha_{ij} = p(e_{ij})$ $e_{ij} = a(s_{j-1}, h_i)$
Decode	$s_j = f(s_{j-1}, y_{j-1}, c)$	$s_j = f(s_{j-1}, y_{j-1}, c_j)$
Generate	$y_j = g(y_{j-1}, s_j, c)$	$y_j = g(y_{j-1}, s_j, c_j)$

$x = (x_1, \dots, x_T)$: input sequence, T : length of input sequence, h_i : hidden states of encoder, c : context vector, α_{ij} : attention weights over input, s_j : decoder hidden state, y_j : output token, f, g : non-linear functions, a : alignment function, p : distribution function

Table 1. Encoder-decoder architecture: traditional and with attention model



Encoder 与 Decoder 的结构差异

模块	Encoder	Decoder
输入类型	源序列（如英文）	目标序列（如法语）
输入处理	词向量 + 位置编码	词向量 + 位置编码
核心机制	自注意力 + 前馈网络	自注意力 + 编码器-解码器注意力 + 前馈网络
注意力机制	仅自注意力（Self-Attention）	三层注意力：Masked 自注意力、Encoder-Decoder 注意力、前馈网络
输出目标	提取上下文特征	生成目标序列

其对齐函数有很多 如下：

$$A(q, K, V) = \sum_i p(a(k_i, q)) * v_i$$

Function	Equation	References
similarity	$a(k_i, q) = \text{sim}(k_i, q)$	[Graves et al. 2014a]
dot product	$a(k_i, q) = q^T k_i$	[Luong et al. 2015a]
scaled dot product	$a(k_i, q) = \frac{q^T k_i}{\sqrt{d_k}}$	[Vaswani et al. 2017]
general	$a(k_i, q) = q^T W k_i$	[Luong et al. 2015a]
biased general	$a(k_i, q) = k_i(W q + b)$	[Sordoni et al. 2016]
activated general	$a(k_i, q) = \text{act}(q^T W k_i + b)$	[Ma et al. 2017b]
generalized kernel	$a(k_i, q) = \phi(q)^T \phi(k_i)$	[Choromanski et al. 2021]
concat	$a(k_i, q) = w_{imp}^T \text{act}(W[q; k_i] + b)$	[Luong et al. 2015a]
additive	$a(k_i, q) = w_{imp}^T \text{act}(W_1 q + W_2 k_i + b)$	[Bahdanau et al. 2015]
deep	$a(k_i, q) = w_{imp}^T E^{(L-1)} + b^L$ $E^{(l)} = \text{act}(W_l E^{(l-1)} + b^l)$ $E^{(1)} = \text{act}(W_1 k_i + W_0 q) + b^1$	[Pavlopoulos et al. 2017]
location-based	$a(k_i, q) = a(q)$	[Luong et al. 2015a]
feature-based	$a(k_i, q) = w_{imp}^T \text{act}(W_1 \phi_1(K) + W_2 \phi_2(K) + b)$	[Li et al. 2019a]

$a(k_i, q)$: alignment function for query q and key k_i , sim : similarity functions such as cosine, d_k : length of input, $(W, w_{imp}, W_0, W_1, W_2)$: trainable parameters, b : trainable bias term, act : activation function.

Table 2. Summary of Alignment Functions

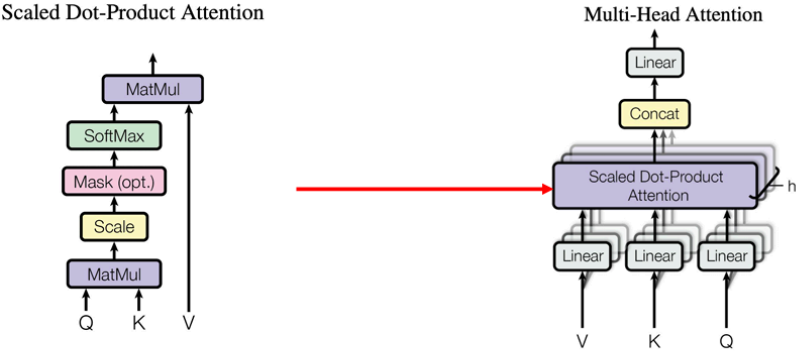
自注意力机制：

Scaled Dot-Product Attention就是一个对齐函数，多头注意力机制是使用h个不同或者相同的对齐函数去更好的理解语义。再加入一个全连接层和线性层就是Multi-Head Attention.

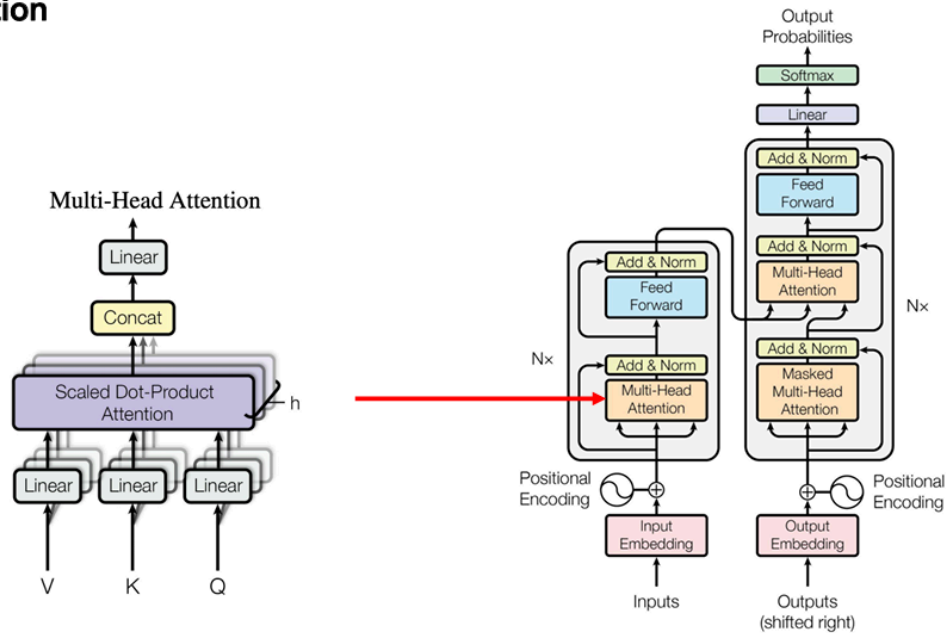
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$$

where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$

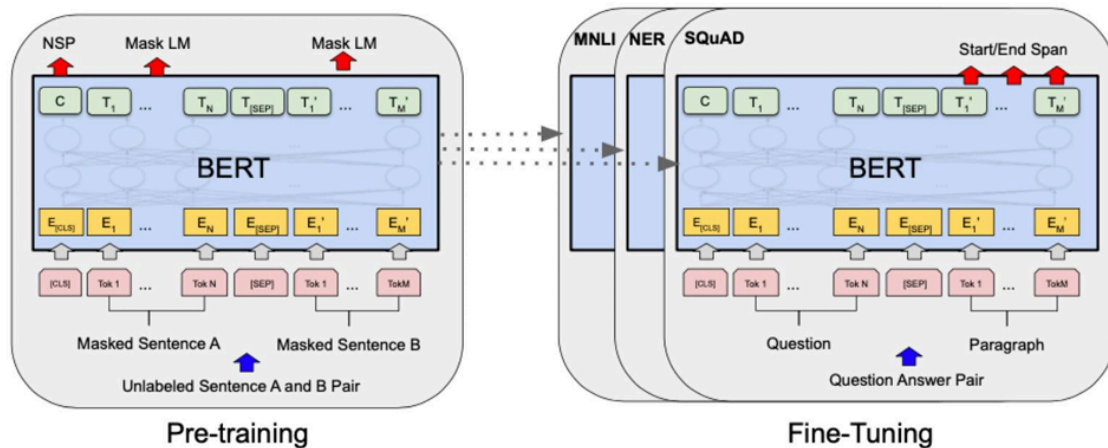


Self-attention



Bert

Bert提出了一个预训练+微调的范式：



1.全方位上下文理解：与以前的模型（例如GPT）相比，BERT能够双向理解上下文，即同时考虑一个词的左边和右边的上下文。这种全方位的上下文理解使得BERT能够更好地理解语言，特别是在理解词义、消歧等复杂任务上有明显优势。

2.预训练+微调（Pre-training + Fine-tuning）的策略：BERT模型先在大规模无标签文本数据上进行预训练，学习语言的一般性模式，然后在具体任务的标签数据上进行微调。这种策略让BERT能够在少量标签数据上取得很好的效果，大大提高了在各种NLP任务上的表现。

3.跨任务泛化能力：BERT通过微调可以应用到多种NLP任务中，包括但不限于文本分类、命名实体识别、问答系统、情感分析等。它的出现极大地简化了复杂的NLP任务，使得只需一种模型就能处理多种任务。

4.多语言支持：BERT提供了多语言版本（Multilingual BERT），可以支持多种语言，包括但不限于英语、中文、德语、法语等，使得NLP任务能够覆盖更广的语言和区域。

在BERT的预训练阶段，BERT通过预测被掩盖的词（masked tokens）来学习双向上下文表示。然而，在微调（fine-tuning）阶段，输入数据通常是完整的句子，不会出现[MASK]标记。这会导致一个潜在的问题：

问题：模型在预训练阶段经常看到[MASK]标记，但在微调阶段从未见过[MASK]标记，这可能会导致预训练和微调之间的一致性。

为了解决这个问题，BERT引入了一种更灵活的掩码策略，避免模型过度依赖[MASK]标记。

在BERT的预训练过程中，掩码策略的具体实现如下：

1. 选择 15% 的词作为目标词：
 - 首先从输入序列中随机选择 15% 的词作为需要预测的目标词。
2. 对目标词进行处理：
 - 对于选中的目标词，按照以下比例进行处理：
 - 80% 的时间：用 [MASK] 替换目标词。
 - 10% 的时间：用随机词替换目标词。

- 10% 的时间：保持目标词不变。

3. 训练目标：

- 模型的任务是根据上下文预测这些目标词的真实值，无论它们是否被替换为 [MASK] 或随机词。

<div><div>BERT vs GPT 差异</div></div>		
特性	BERT	GPT
训练方式	自编码 (Autoencoding)	自回归 (Autoregressive)
预测目标	给定上下文，预测其中的一个或多个缺失单词	在给定前面的单词时，预测下一个单词
输入处理	双向，可以同时考虑一个词的左右上下文	单向 (从左到右或者从右到左)
适用场景	适合理解上下文，有助于信息提取、问答系统、情感分析等	适合生成式任务，如文章生成、诗歌创作等
架构	基于Transformer的编码器	基于Transformer的解码器
语言模型	判别式 (Discriminative)	生成式 (Generative)
优点	对上下文理解能力较强	预测的连贯性较强
缺点	生成的文本连贯性较弱	对上下文理解能力相对较弱