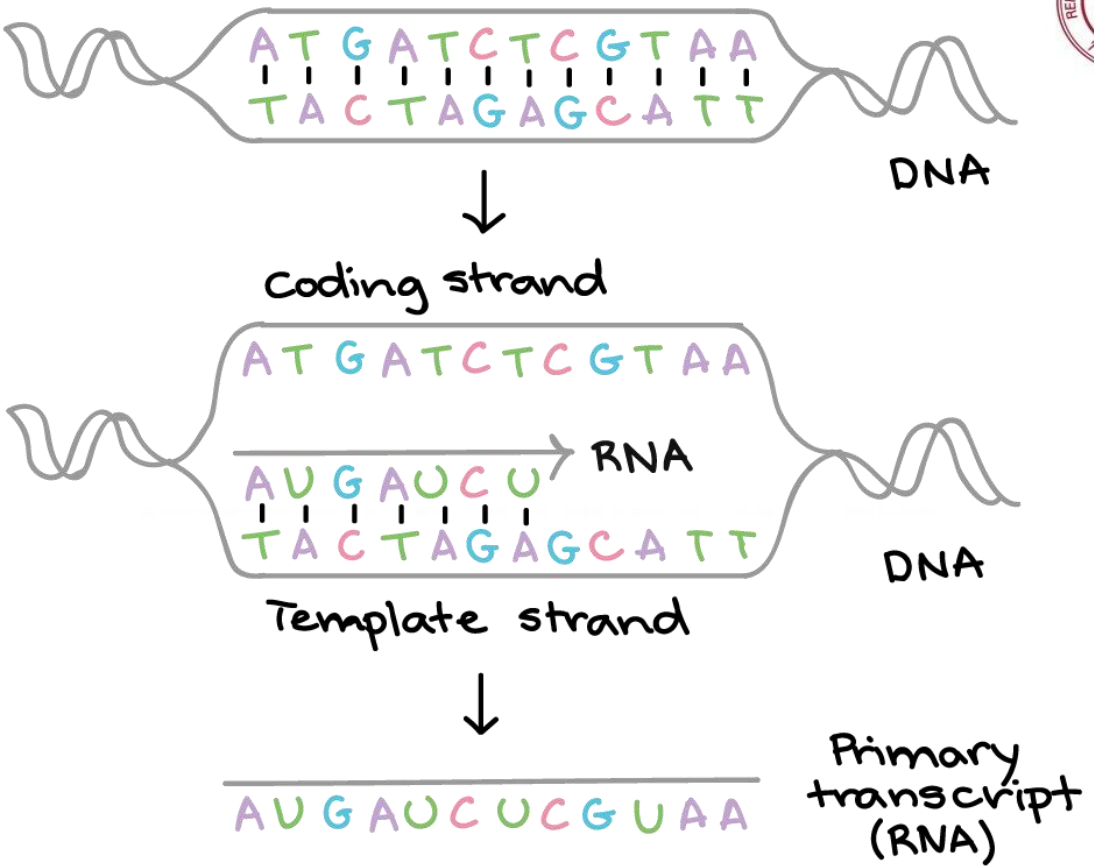
Transformers

- Beyond Recurrent Neural Network

概要

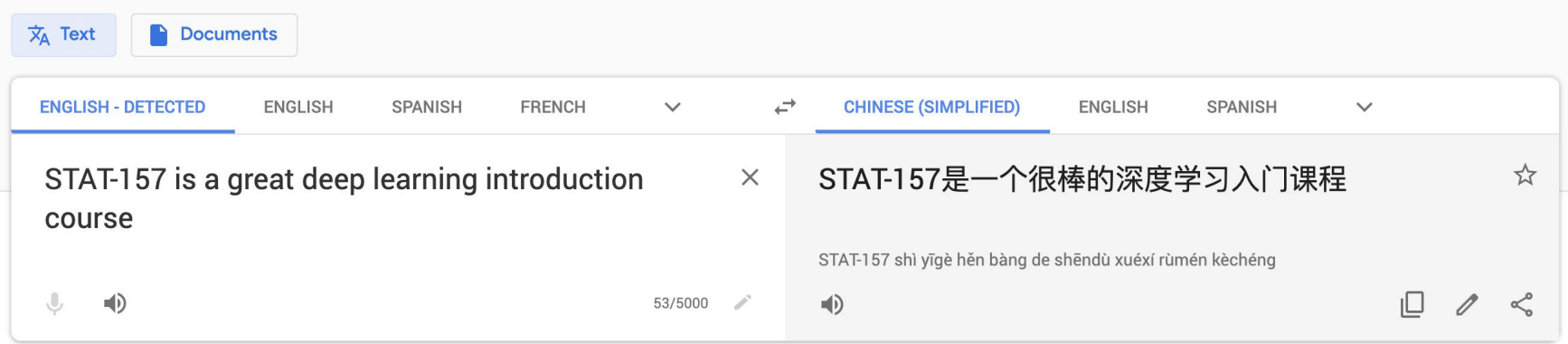
- Seq2Seq模型
- 注意力机制
- Transformers模型
- BERT模型

Seq2seq
模型


ATGATCTCGTAA
TACTAGAGCATT
DNA
coding strand
ATGATCTCGTAA
AUGAUCU
RNA
TACTAGAGCATT
DNA
Template strand
AUGAUCUCGUAA
Primary transcript (RNA)

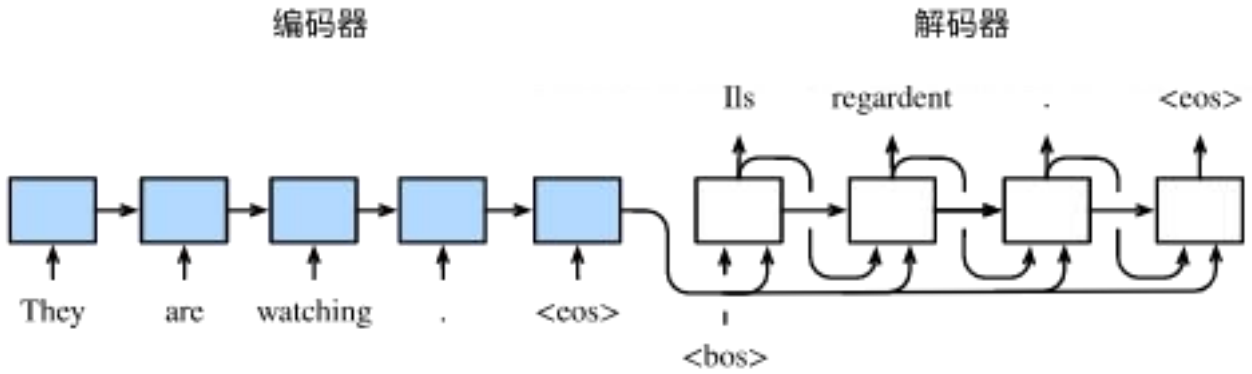
机器翻译

- 给定源语言中的句子，翻译成目标语言
- 这两个序列可以具有不同的长度



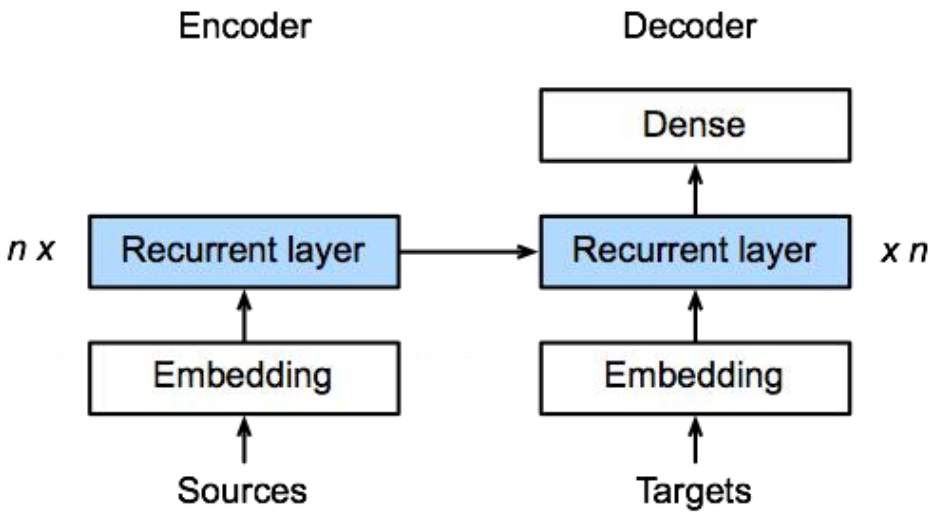
Seq2seq 模型

- 编码器是读取输入序列的 RNN
- 解码器使用另一个 RNN 来生成输出



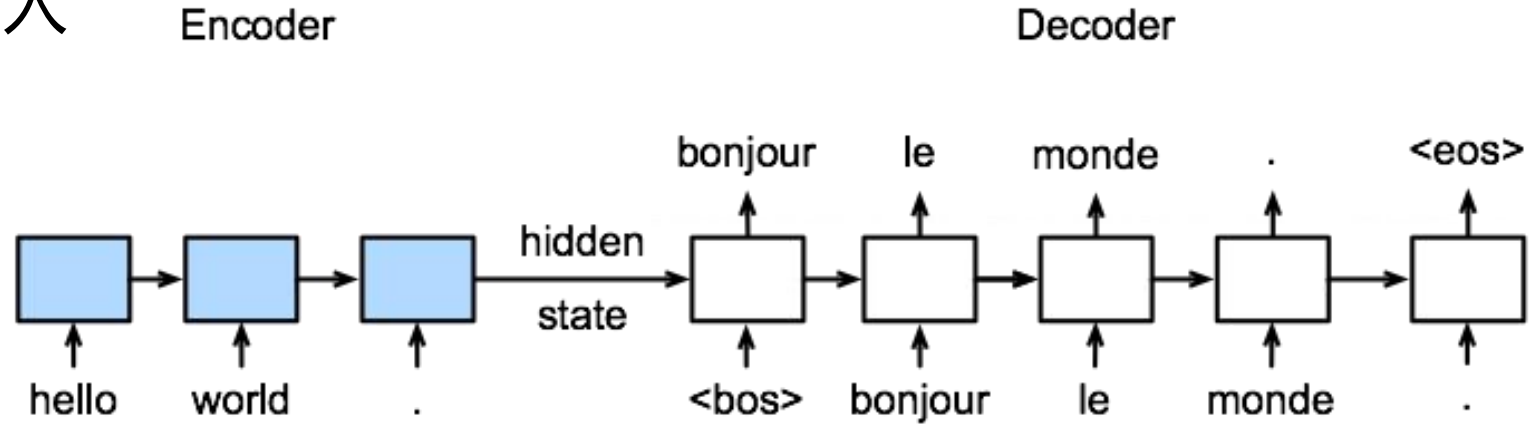
Seq2seq 模型

- 编码器是没有输出层的标准 RNN 模型
- 编码器在最后时间步骤中的隐含状态用作解码器的初始隐藏状态

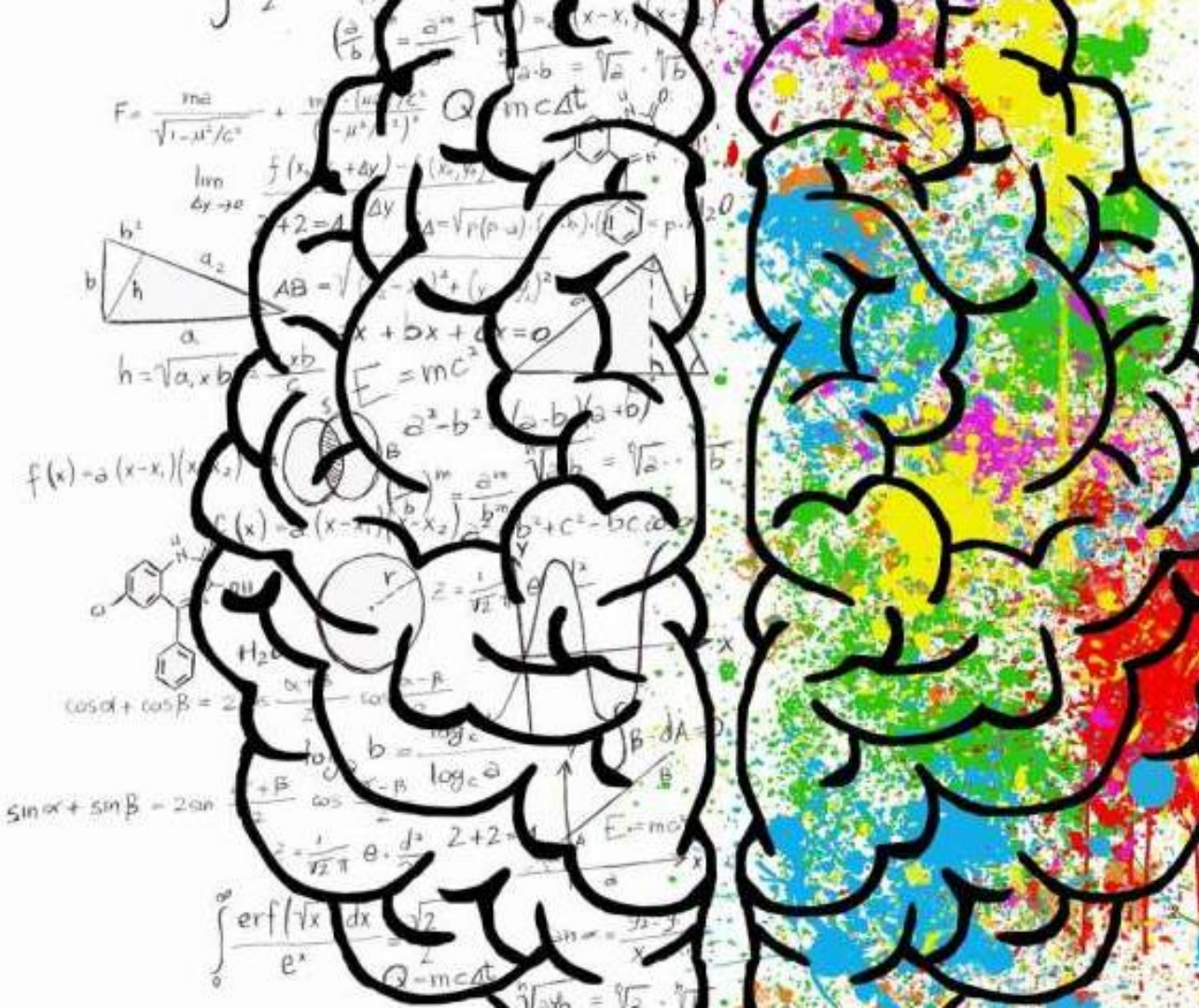


机器翻译训练

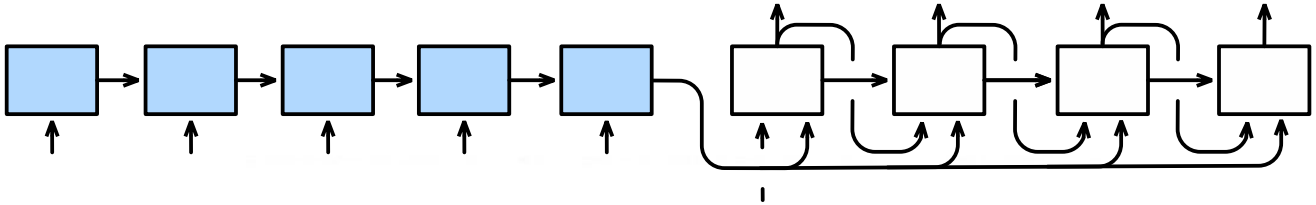
- 在训练期间，解码器（Decoder）用目标语言句子作为输入



注意力机制

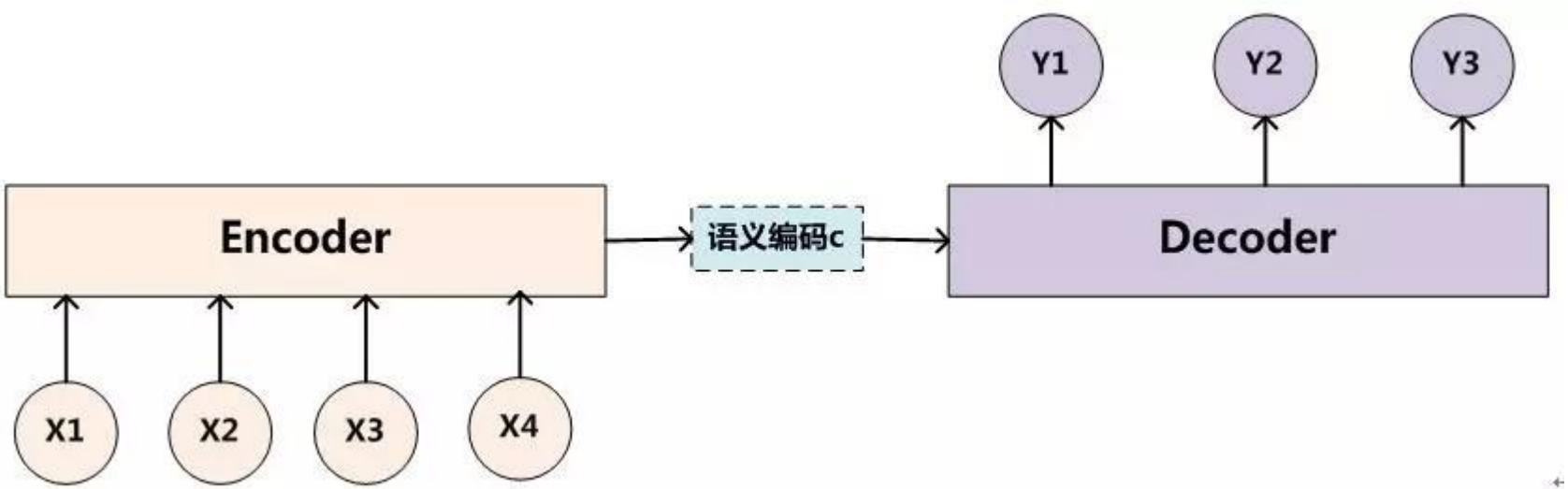


改进传统RNN



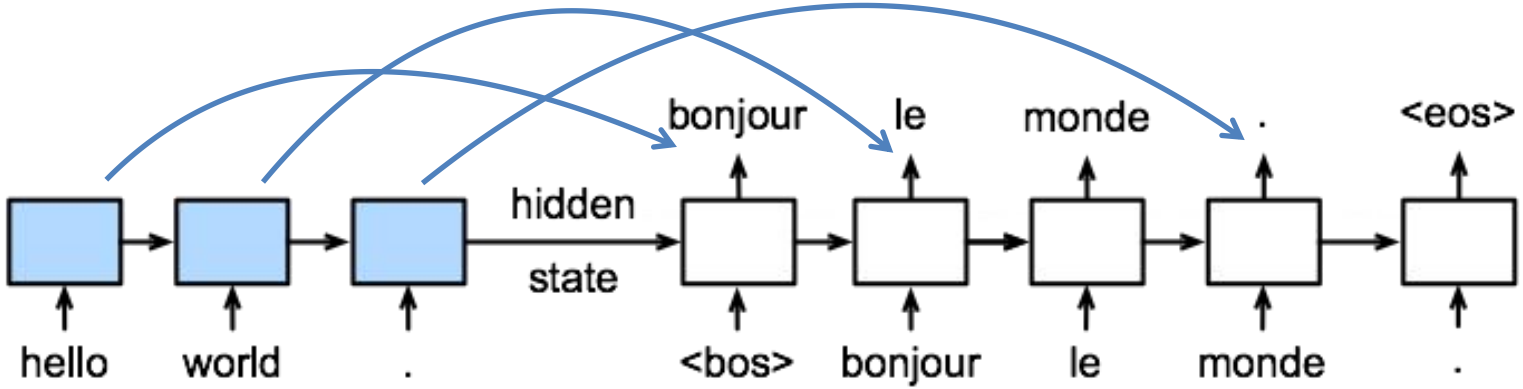
动机

- 每个生成的token可能与不同的源token相关



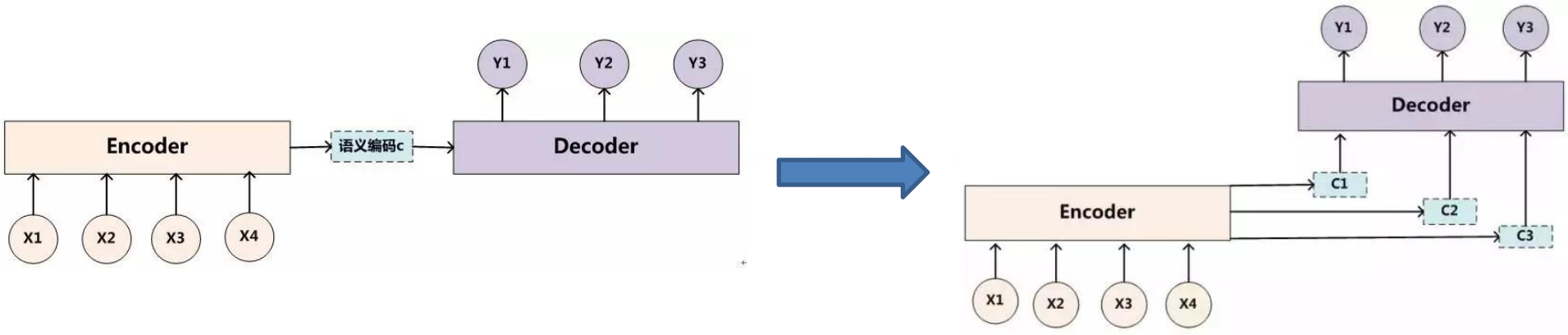
动机

- 每个生成的token可能与不同的源token相关



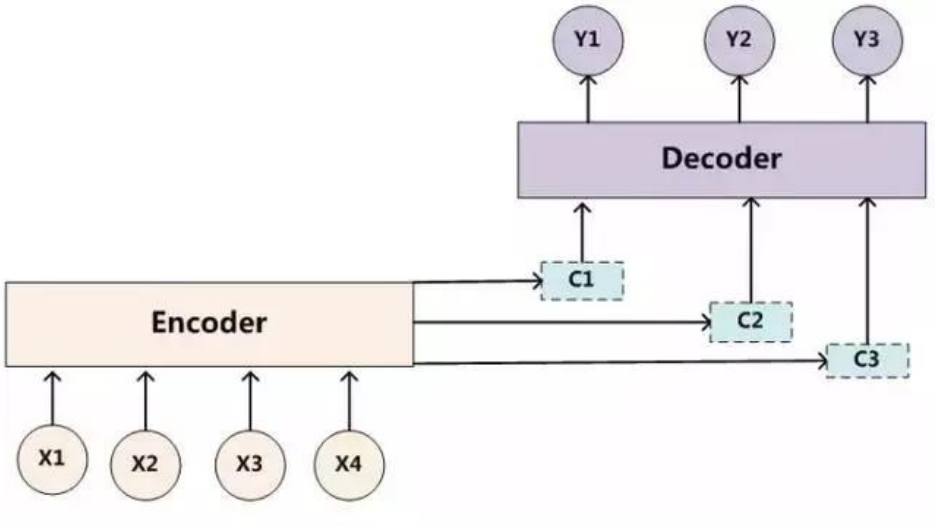
动机

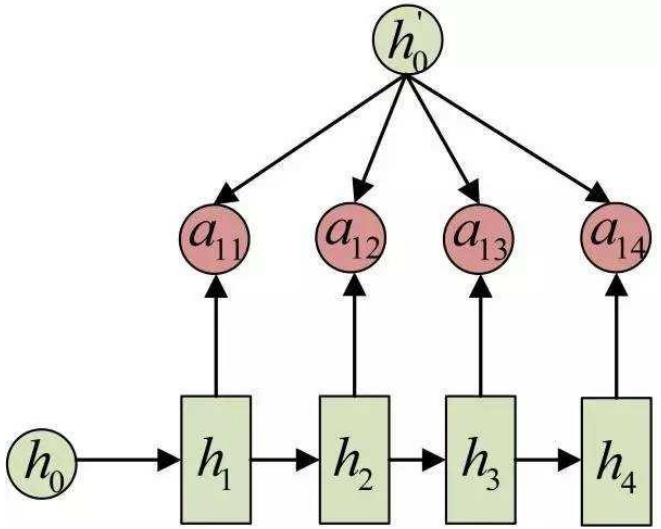
- 每个生成的token可能与不同的源token相关



动机

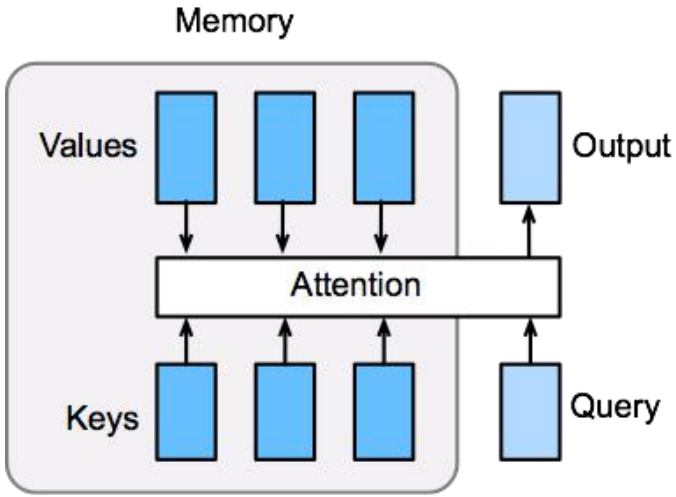
- 每个生成的token可能与不同的源token相关





注意力层

- 注意力层明确选择相关信息
  - 它的存储器（memory）由“键值对”组成
  - 键和查询越相似，则输出的值越相近



注意力层

- 假设“一条询问”为 $\mathbf{q} \in \mathbb{R}^{d_q}$ ，存储器为 $(\mathbf{k}_1, \mathbf{v}_1), \ldots, (\mathbf{k}_n, \mathbf{v}_n)$ ；$\mathbf{k}_i \in \mathbb{R}^{d_k}$ ，$\mathbf{v}_i \in \mathbb{R}^{d_v}$
- 计算 n 分数 $a_1, \ldots, a_n$ ；$a_i = \alpha(\mathbf{q}, \mathbf{k}_i)$
- 使用 softmax 获得注意力

$$b_1, \ldots, b_n = \mathrm{softmax}(a_1, \ldots, a_n)$$

- 输出是值的加权和

$$\mathbf{o} = \sum_{i=1}^{n} b_i \mathbf{v}_i$$

改变α可以获得不同的注意力层

点乘注意力

- 假设询问的长度与值相同 $\mathbf{q}, \mathbf{k}_i \in \mathbb{R}^d$

$$\alpha(\mathbf{q}, \mathbf{k}) = \langle \mathbf{q}, \mathbf{k} \rangle / \sqrt{d}$$

- 向量化版本
  - m 个询问 $\mathbf{Q} \in \mathbb{R}^{m \times d}$ 和 n 个键 $\mathbf{K} \in \mathbb{R}^{n \times d}$

$$\alpha(\mathbf{Q}, \mathbf{K}) = \mathbf{Q}\mathbf{K}^T / \sqrt{d}$$

多层感知注意力

- 可学习的参数 $\mathbf{W}_k \in \mathbb{R}^{h \times d_k}, \mathbf{W}_q \in \mathbb{R}^{h \times d_q}, \mathbf{v} \in \mathbb{R}^h$

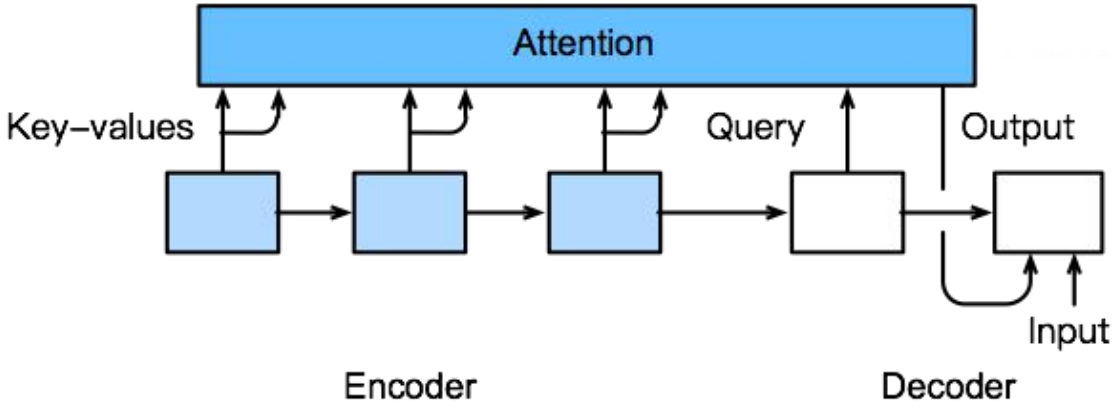
$$\alpha(\mathbf{k}, \mathbf{q}) = \mathbf{v}^T \tanh(\mathbf{W}_k \mathbf{k} + \mathbf{W}_q \mathbf{q})$$

- 相当于连接“键”（key）和“询问”（query），然后输入隐含大小为 h 和输出大小 1的单个隐含层感知

Seq2seq 与注意力机制

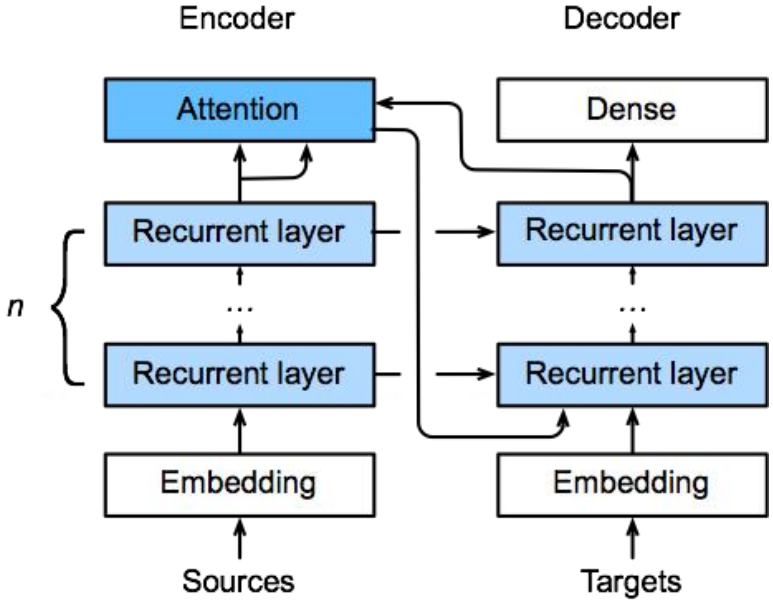
模型架构

- 添加额外的注意层以编码器的输出作为存储器
- 注意力的输出用作解码器的输入



编码器—解码器上的注意力机制

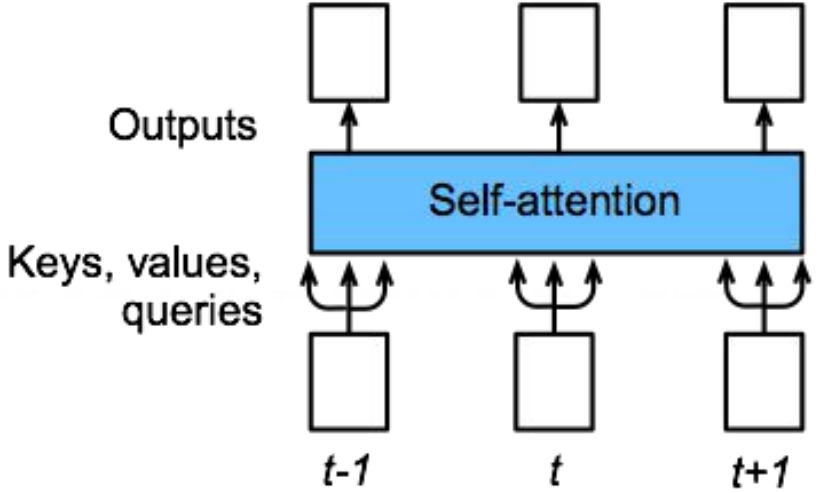
- 使用编码器中最后一个循环神经网络层的输出
- 然后，注意力输出与嵌入输出拼接，以输入解码器中的第一个循环神经网络层



变换器模型
(Transformer)

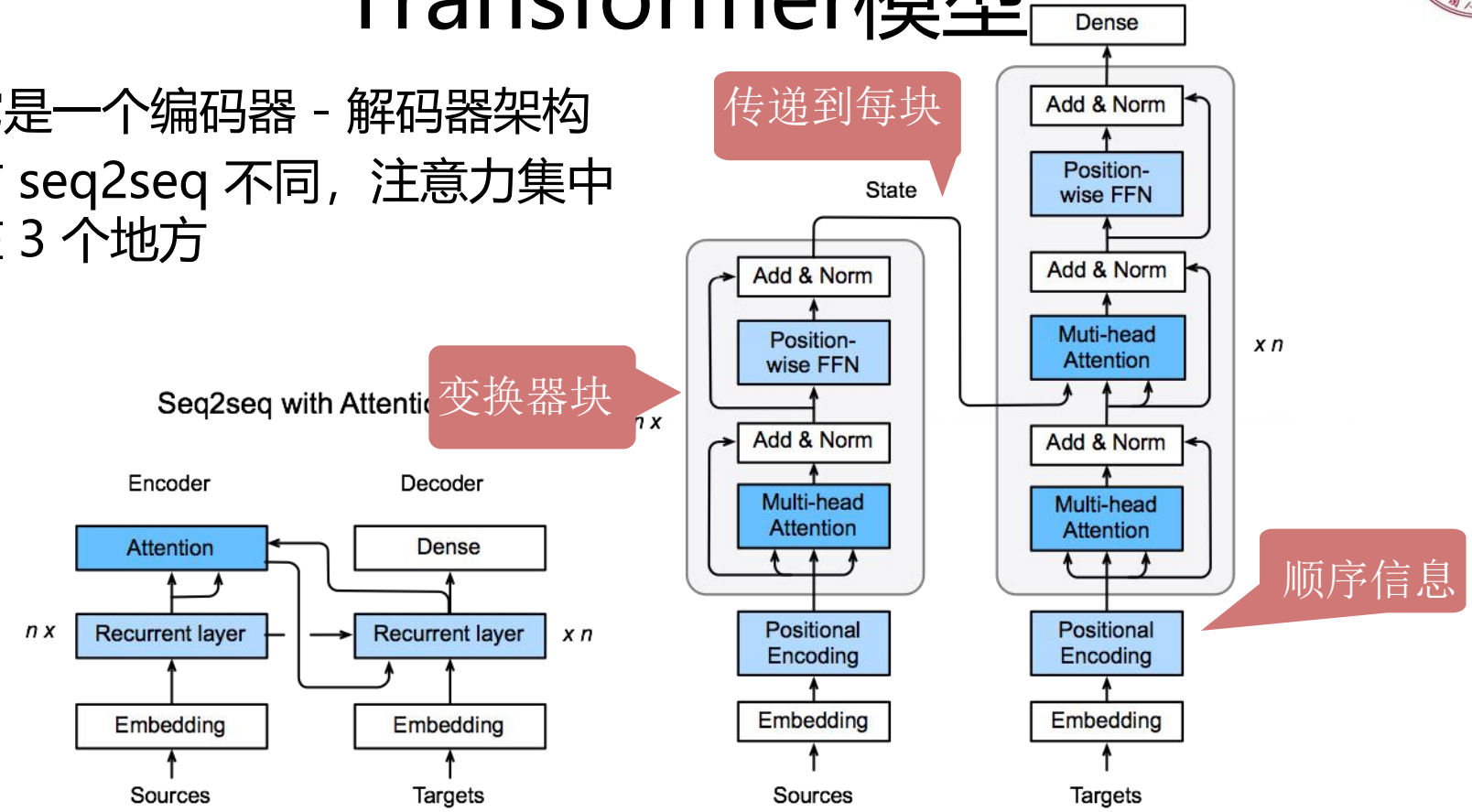
自注意力机制

- 要使用 n 个输入生成 n 个输出，我们可以将每个输入复制为键（key），值（value）和查询（query）中
- 不保留顺序信息
- 并行计算

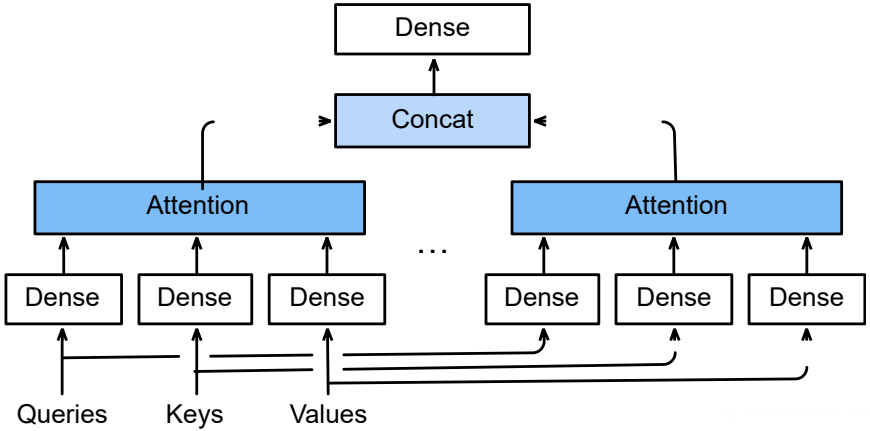


Transformer模型

- 它是一个编码器 - 解码器架构
- 与 seq2seq 不同，注意力集中在 3 个地方



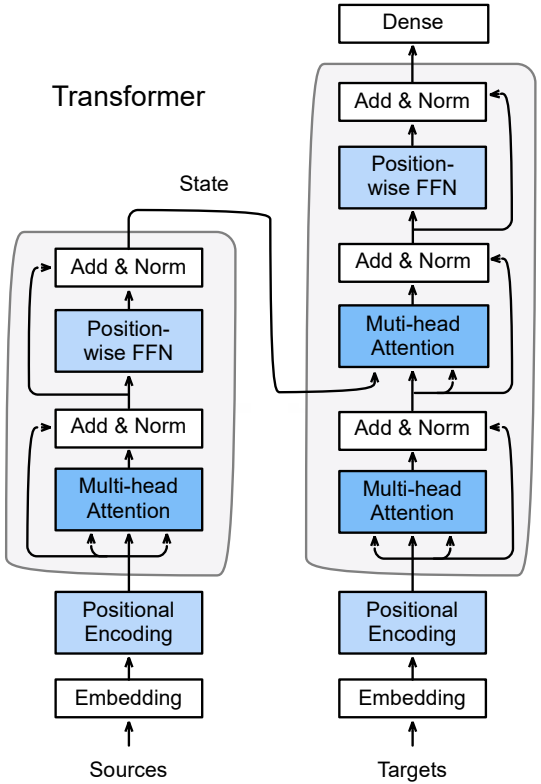
多头注意力机制（Multi-head Attention）



$$\mathbf{W}_q^{(i)} \in \mathbb{R}^{p_q \times d_q}, \mathbf{W}_k^{(i)} \in \mathbb{R}^{p_k \times d_k}, \text{ and } \mathbf{W}_v^{(i)} \in \mathbb{R}^{p_v \times d_v}$$

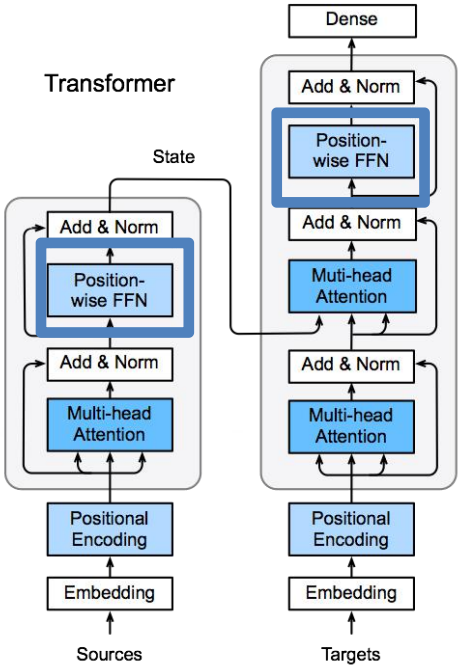
$$\mathbf{o}^{(i)} = \text{attention}(\mathbf{W}_q^{(i)}\mathbf{q}, \mathbf{W}_k^{(i)}\mathbf{k}, \mathbf{W}_v^{(i)}\mathbf{v})$$

$$\text{for } i = 1, \ldots, h \qquad \mathbf{o} = \mathbf{W}_o \begin{bmatrix} \mathbf{o}^{(1)} \\ \vdots \\ \mathbf{o}^{(h)} \end{bmatrix}$$



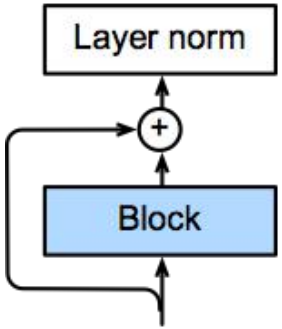
位置前馈网络

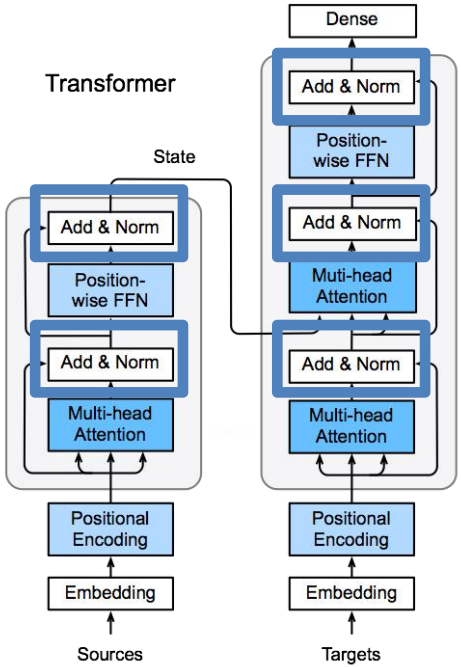
- 将输入（批量大小，序列长度，特征集大小）重新整形为（批量*序列长度，特征集大小）
- 用两层 MLP
- 转换为 3-D 形态
- 等于应用两（1,1）个卷积层



添加与归一化

- 层规范（Layer Norm）类似于批量规范（Batch Norm）
- 但是平均值和方差是沿最后一个维度计算的
  - X.mean（axis = -1）而不是批量归一化
  - X.mean 中的第一个批次维度（axis = 0）





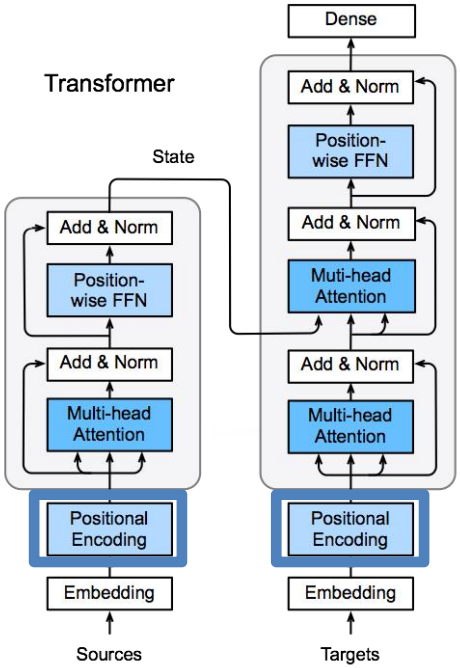
位置编码

- 假设嵌入 $X \in \mathbb{R}^{l \times d}$ 输出的形状（序列长度，嵌入维度）
- 创建 $P \in \mathbb{R}^{l \times d}$

$$P_{i,2j} = \sin(i/10000^{2j/d})$$

$$P_{i,2j+1} = \cos(i/10000^{2jd})$$

- 输出 $X + P$



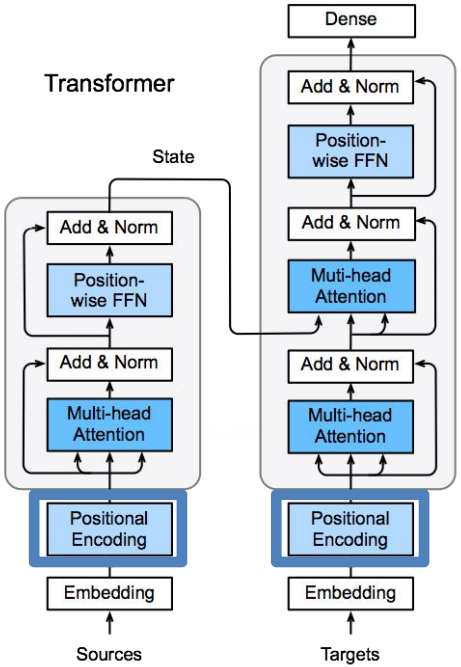
位置编码

$$\begin{cases} PE(pos, 2i) = \sin\left(pos/10000^{2i/d_{model}}\right) \\ PE(\text{pos}, 2i+1) = \cos\left(pos/10000^{2i/d_{model}}\right) \end{cases}$$

$$\begin{cases} sin(\alpha+\beta) = sin\alpha cos\beta + cos\alpha sin\beta \\ cos(\alpha+\beta) = cos\alpha cos\beta - sin\alpha sin\beta \end{cases}$$

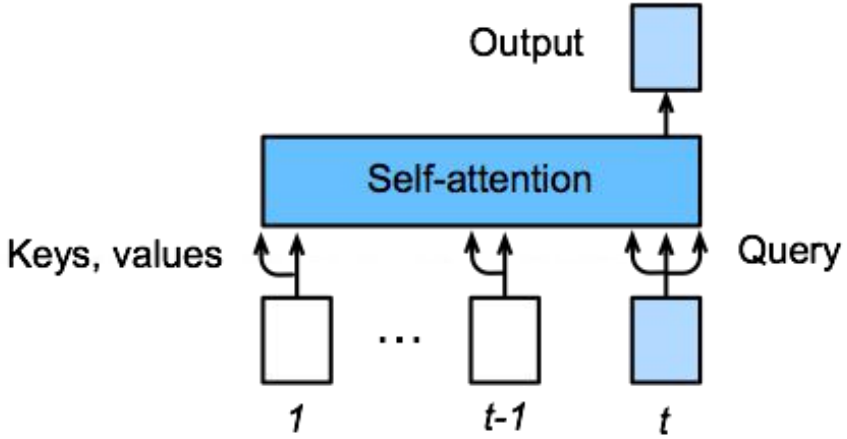
我们可以得到：

$$\begin{cases} PE(pos+k, 2i) = PE(pos, 2i) \times PE(k, 2i+1) + PE(pos, 2i+1) \times PE(k, 2i) \\ PE(pos+k, 2i+1) = PE(pos, 2i+1) \times PE(k, 2i+1) - PE(pos, 2i) \times PE(k, 2i) \end{cases}$$

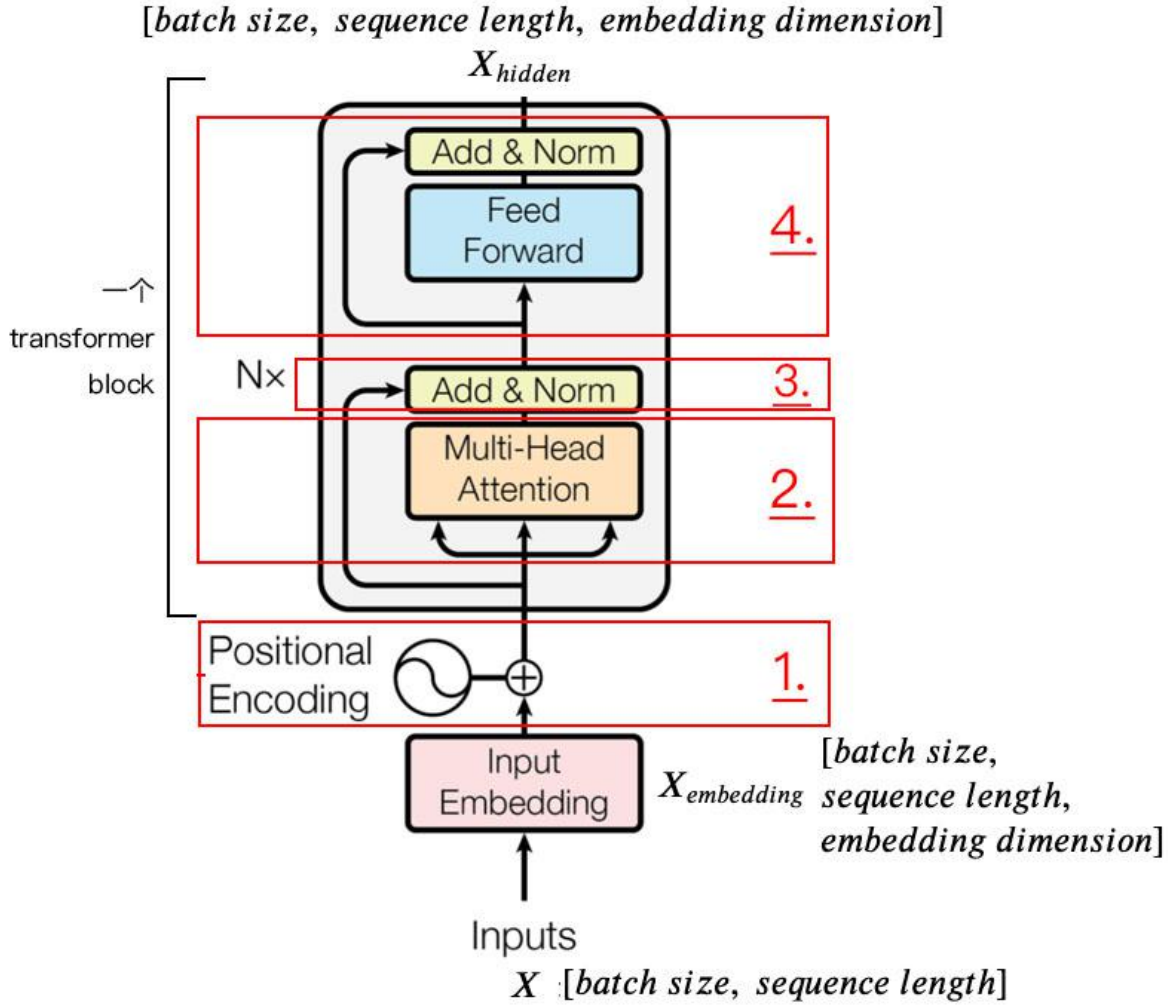


预测

- 在时刻 t 预测：
  - 用之前输入的键和值
  - 时刻 t 输入作为查询，以及键和值，以预测输出

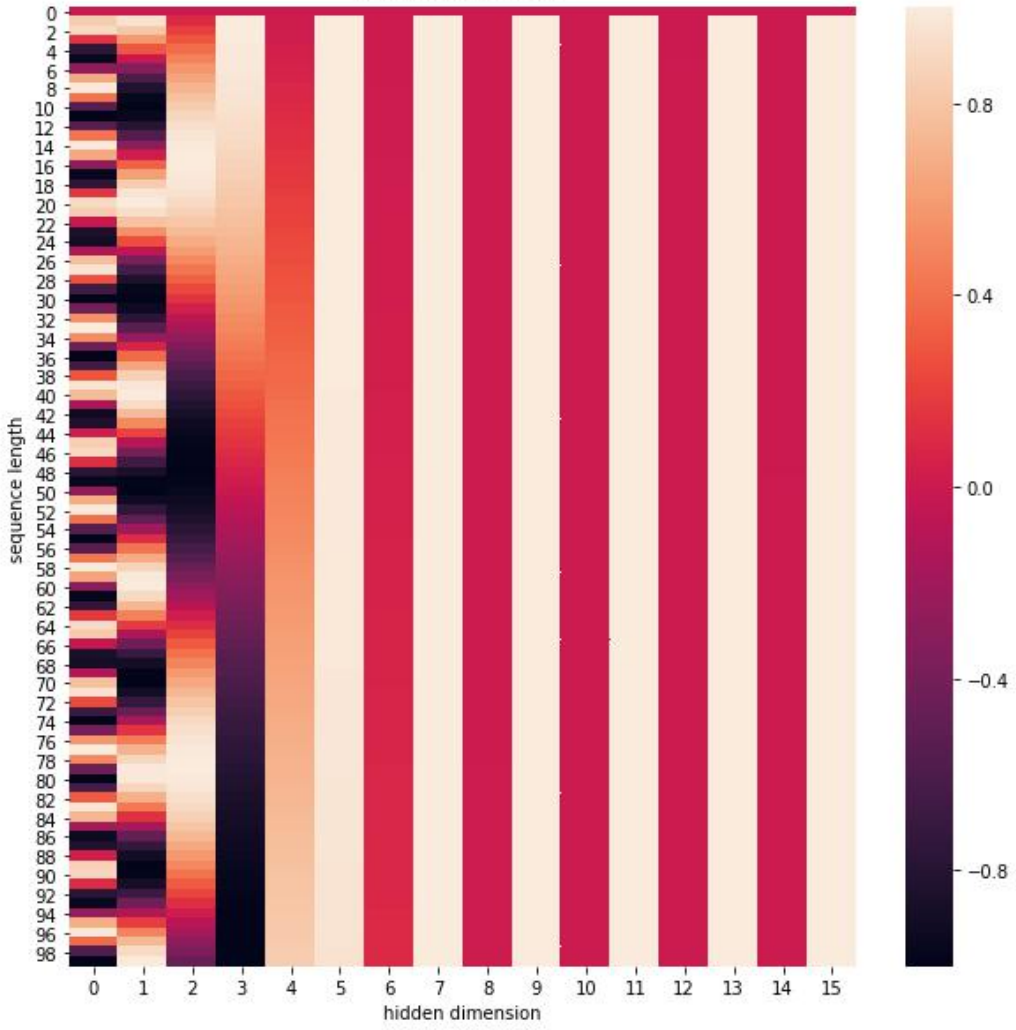


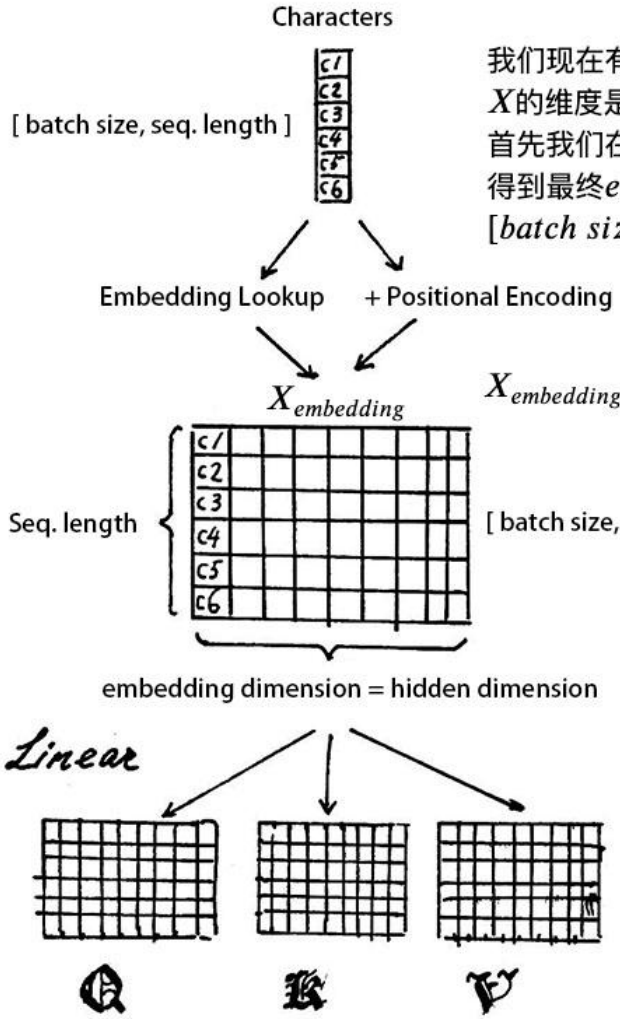
从自然语言序列
经过计算
到隐藏层的过程



$$\begin{cases} PE(pos, 2i) = \sin\left(pos/10000^{2i/d_{model}}\right) \\ PE(\text{pos}, 2i+1) = \cos\left(pos/10000^{2i/d_{model}}\right) \end{cases}$$

上式中pos指的是句中字的位置, 取值范围是[0, max sequence length), i指的是词向量的维度, 取值范围是[0, embedding dimension), 上面有sin和cos一组公式, 也就是对应着embedding dimension维度的一组奇数和偶数的序号的维度, 例如0,1一组, 2,3一组, 分别用上面的sin和cos函数做处理, 从而产生不同的周期性变化, 而位置嵌入在embedding dimension维度上随着维度序号增大, 周期变化会越来越慢, 而产生一种包含位置信息的纹理, 就像论文原文中第六页讲的, 位置嵌入函数的周期从2π到10000* 2π变化, 而每一个位置在embedding dimension维度上都会得到不同周期的sin和cos函数的取值组合, 从而产生独一的纹理位置信息, 模型从而学到位置之间的依赖关系和自然语言的时序特性.





我们现在有了词向量矩阵和位置嵌入，假如我们现在有一些句子X，
X的维度是$[batch\ size,\ sequence\ length]$，
首先我们在字向量里查到相应的嵌入，然后与位置嵌入元素相加，
得到最终$embedding$的维度为：
$[batch\ size,\ sequence\ length,\ embedding\ dimension]$

$$X \in \mathbb{R}^{batch\ size * seq.\ len.}$$

$$X_{embedding} = EmbeddingLookup(X) + PositionalEncoding$$

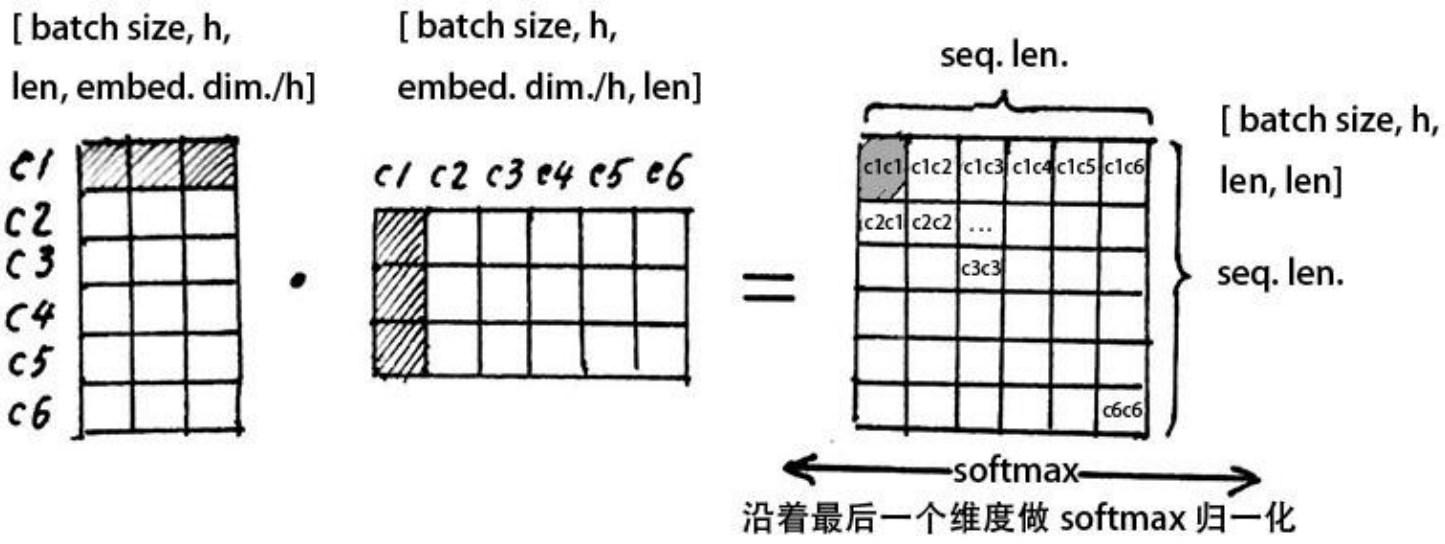
$$X_{embedding} \in \mathbb{R}^{batch\ size * seq.\ len. * embed.\ dim.} \quad \text{(eq.2)}$$

下一步，为了学到多重含义的表达，
对$X_{embedding}$做线性映射，
也就是分配三个权重，

$$W_Q, W_K, W_V \in \mathbb{R}^{embed.dim. * embed.dim.},$$

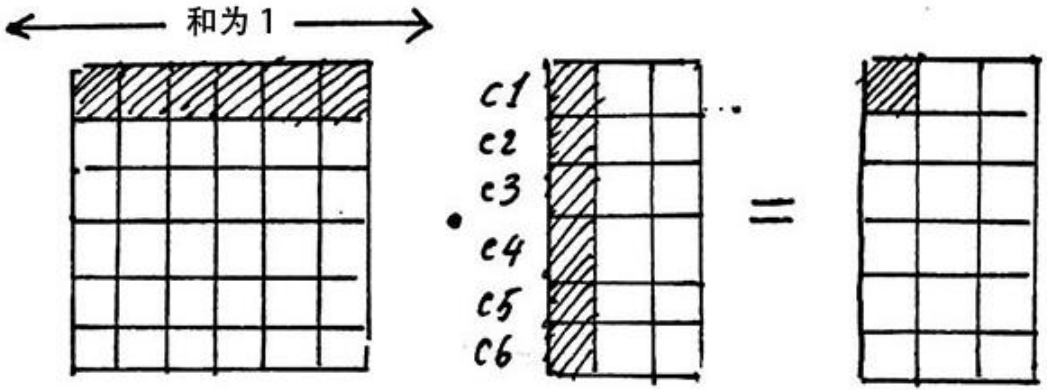
线性映射之后，形成三个矩阵，为Q，K，V，
和线性变换之前的维度一致.

$$Q = Linear(X_{embedding}) = X_{embedding} W_Q$$
$$K = Linear(X_{embedding}) = X_{embedding} W_K \quad \text{(eq.3)}$$
$$V = Linear(X_{embedding}) = X_{embedding} W_V$$


[batch size, h,
len, embed. dim./h]
[batch size, h,
embed. dim./h, len]
seq. len.
[batch size, h,
len, len]
seq. len.
c1 c2 c3 c4 c5 c6
c1c1 c1c2 c1c3 c1c4 c1c5 c1c6
c2c1 c2c2 ...
c3c3
c6c6
softmax
沿着最后一个维度做 softmax 归一化

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \quad \text{(eq. 4)}$$

上式中就是自注意力机制, 我们先求QK^T, 也就是求出注意力矩阵, 然后用注意力矩阵给V加权,
$\sqrt{d_k}$是为了把注意力矩阵变成标准正态分布, 使得$softmax$归一化之后的结果更加稳定,
以便反向传播的时候获取平衡的梯度, 详见原文第4页角注.



我们目前得到了注意力矩阵, 并用$softmax$归一化, 使**每个字**跟**其他所有字**的注意力权重的和为1,
注意力矩阵的作用就是一个注意力权重的概率分布, 我们要用注意力矩阵的权重给V进行加权,
上图中我们从注意力矩阵取出一行(和为1)然后依次点乘V的列,
矩阵V的每一行代表着每个字向量的数学表达,
我们上面的操作正是用注意力权重进行这些数学表达的加权线性组合,
从而使每个字向量都含有当前句子内所有字向量的信息.

Layer Normalization 和残差连接

1). **残差连接**:

我们在上一步得到了经过注意力矩阵加权之后的V, 也就是$Attention(Q,\ K,\ V)$, 我们对它进行一下转置, 使其和$X_{embedding}$的维度一致, 也就是$[batch\ size,\ sequence\ length,\ embedding\ dimension]$, 然后把他们加起来做残差连接, 直接进行元素相加, 因为他们的维度一致:

$$X_{embedding} + Attention(Q,\ K,\ V)$$

在之后的运算里, 每经过一个模块的运算, 都要把运算之前的值和运算之后的值相加, 从而得到残差连接, 训练的时候可以使梯度直接走捷径反传到最初始层:

$$X + SubLayer(X)$$

2). $LayerNorm$:

$LayerNormalization$的作用是把神经网络中隐藏层归一为标准正态分布, 也就是$i.\ i.\ d$独立同分布, 以起到加快训练速度, 加速收敛的作用:

$$\mu_i = \frac{1}{m}\sum_{i=1}^{m} x_{ij}$$

上式中以矩阵的行(row)为单位求均值;

$$\sigma_j^2 = \frac{1}{m}\sum_{i=1}^{m}(x_{ij} - \mu_j)^2$$

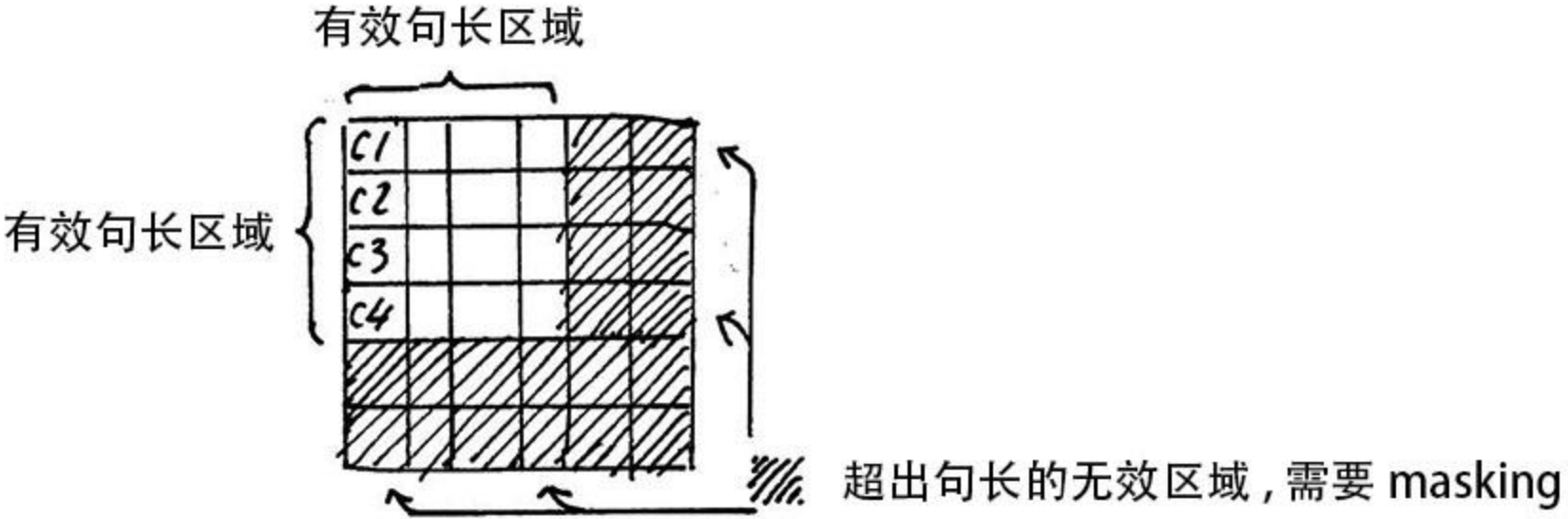
上式中以矩阵的行(row)为单位求方差;

$$LayerNorm(x) = \alpha \odot \frac{x_{ij} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} + \beta$$

然后用**每一行**的**每一个元素**减去**这行的均值**, 再除以**这行的标准差**, 从而得到归一化后的数值, ϵ是为了防止除0;

之后引入两个可训练参数α, β来弥补归一化的过程中损失掉的信息, 注意$\odot$表示元素相乘而不是点积, 我们一般初始化α为全1, 而β为全0.

Attention Mask



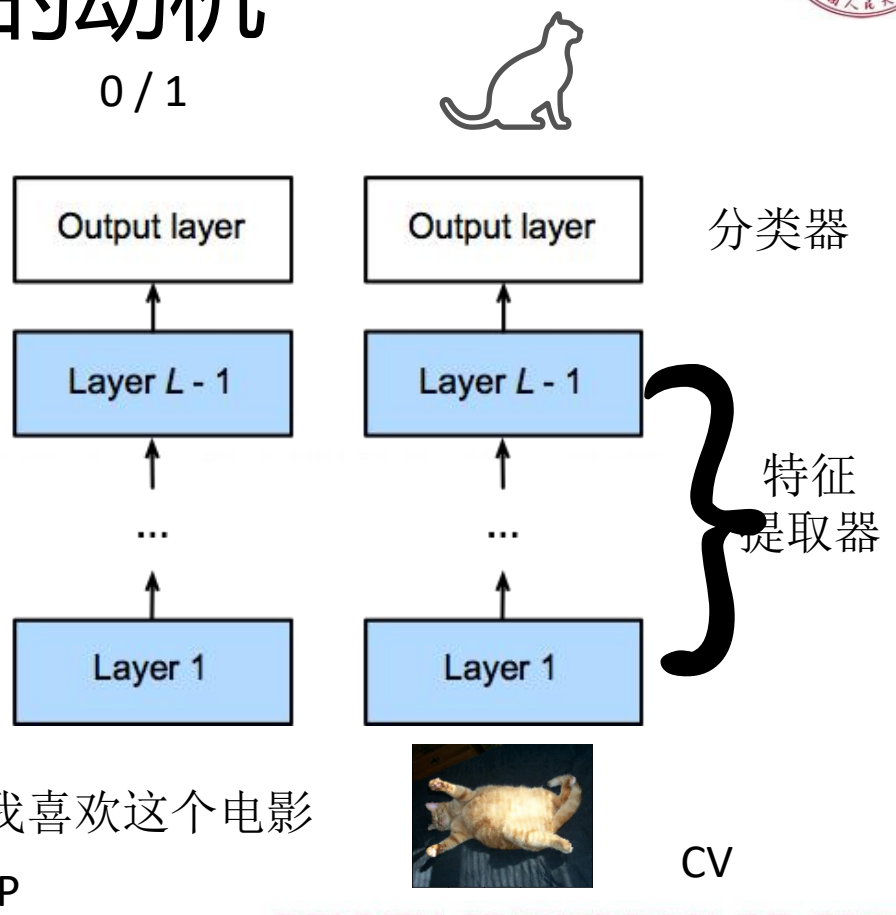
OF CHINA

NLP中的迁移学习（Transfer Learning）

- 使用预先训练的模型为新任务提取单词/句子功能
  - 例如：word2vec 或语言模型
- 通常不更新预先训练的模型的权重
- 需要构建一个新模型来捕获新任务所需的信息
  - Word2vec 忽略顺序信息，这个语言模型只查看单一方向

BERT 的动机

- 基于微调的 NLP 方法
- 预训练模型捕获足够多的数据信息
- 只需要为新任务添加简单的输出层

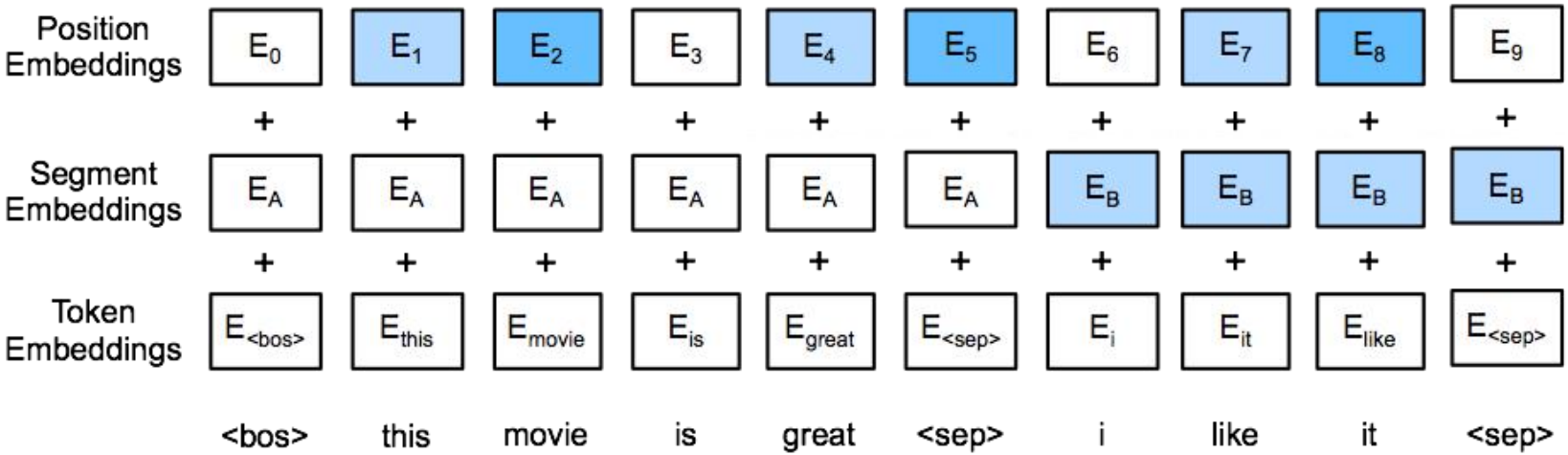


BERT 架构

- 一个（巨大的）变换器模型编码器（没有解码器）
- 两种模型大小：
  - 基础版：＃blocks = 12，隐含大小= 768，＃heads = 12，＃参数 = 110M
  - 增强版：＃blocks = 24，隐含大小= 1024，＃heads = 16，＃参数= 340M
- 使用超过30亿单词的大型语料库（书籍和维基百科）训练

输入

- 每个样本都是一对句子
- 添加其他细分嵌入



预训练任务1：掩码语言模型

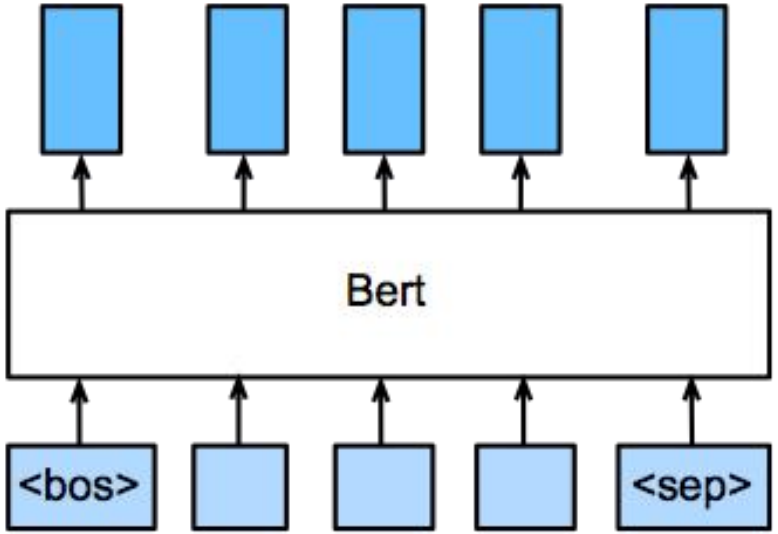
- 在每个句子中随机掩盖（例如 15%）标记，预测这些掩码标记（ <mask>）
  - 变换器模型是双向的，它打破了标准语言模型的单向限制
- 微调任务中没有掩码标记（<mask>）
  - 80% 的时间，用 <mask> 替换选定的标记
  - 10% 的时间，用随机挑选的picked tokens替换
  - 10% 的时间，保留原始标记

预训练任务2：下一句话预测

- 50% 的时间，选择一个连续的句子对
  - <bos> 这部电影很棒 <sep> 我喜欢它 <sep>
- 50% 的时间，选择一个随机的句子对
  - <bos> 这部电影很棒 <sep> hello world <sep>
- 将变换器模型的输出 <bos> 输入到稠密层以预测它是否是顺序对（ sequential pair）

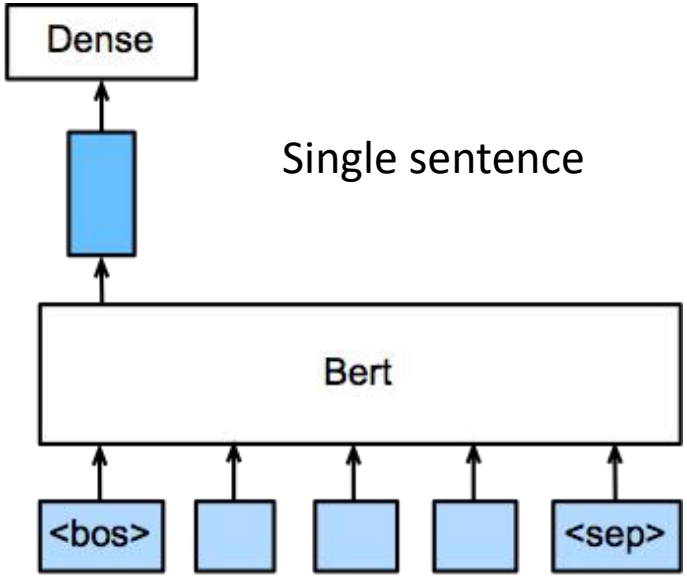
用 Bert 微调

- Bert 为捕获上下文信息的每个标记返回一个特征向量
- 不同的微调任务使用不同的向量集

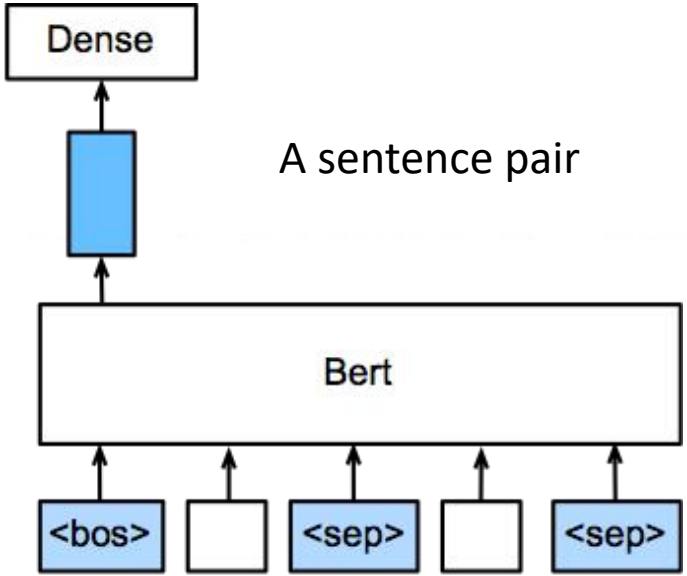


语句分类

- 将 <bos> 标记向量输入稠密输出层



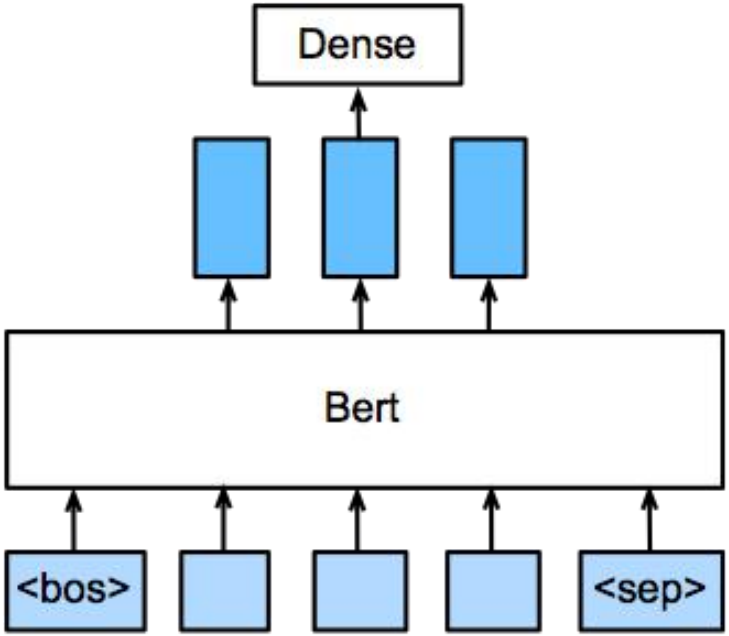
Single sentence



A sentence pair

命名实体识别

- 确定标记是否是命名实体，例如人员，组织和位置等等
- 将每个非特殊标记向量馈送到稠密输出层



自动问答

- 给定问题和描述文本，找到答案，这是描述中的文本段
- 给定 p_i，描述中的第 i 个标记，学习 $\mathbf{s}$ 中的 p_i，第 i 个标记是这段开始的概率：

$$p_1, \ldots, p_T = \text{softmax}(\langle \mathbf{s}, \mathbf{v}_1 \rangle, \ldots, \langle \mathbf{s}, \mathbf{v}_T \rangle)$$

- 同样可以学习第 i 个标记是这段结局的概率