注意力、序列模型、Transformer

内容

- 注意力机制
  - 注意力模型
  - 自注意力模型
  - 多头注意力模型
  - 多头自注意力模型
- 序列生成模型
  - 序列概率模型
  - 束搜索
  - 序列到序列模型
  - 评价方法
- Transformer模型

注意力机制

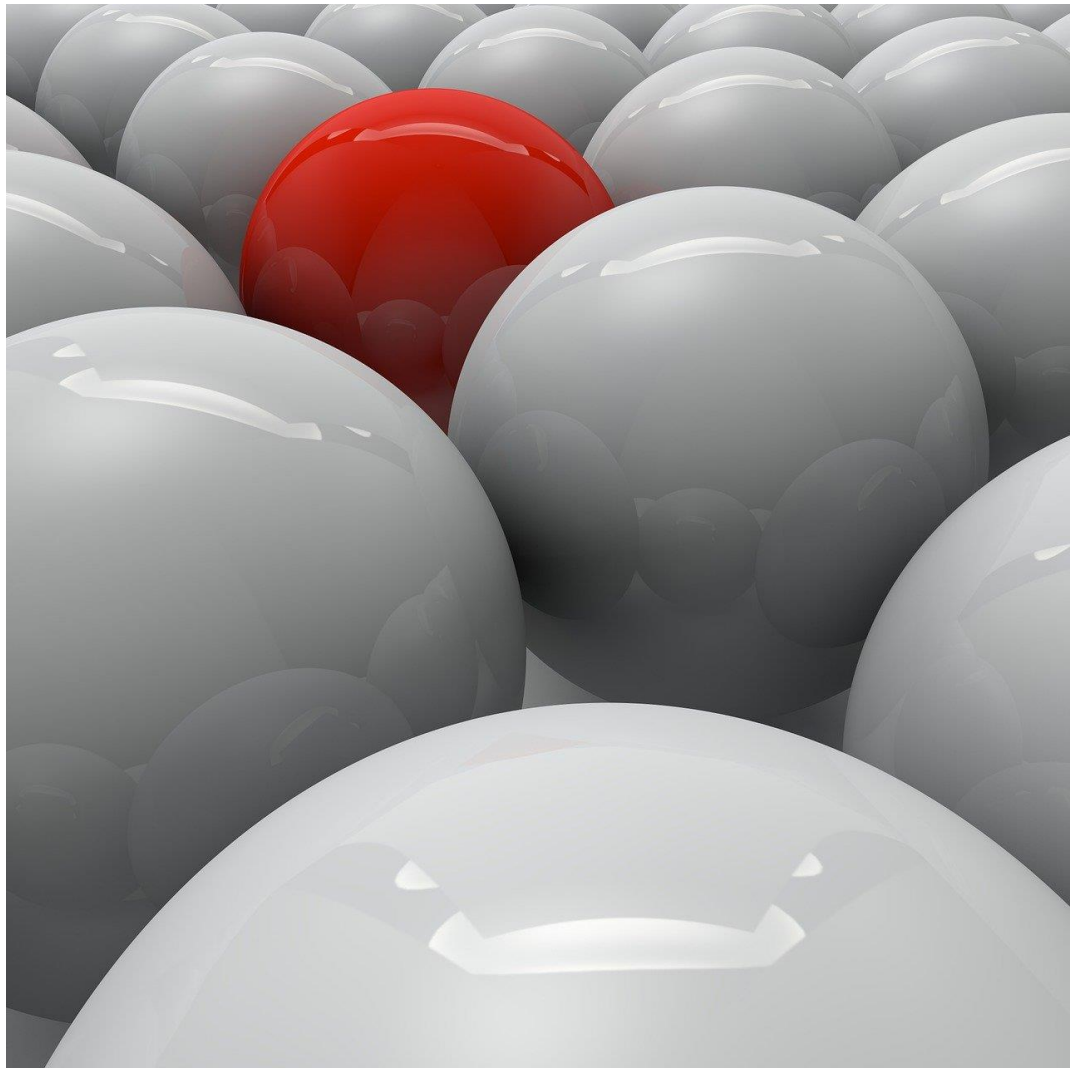
通用近似定理

- 由于优化算法和计算能力的限制，神经网络在实践中很难达到通用近似的能力。
  - 网络不能太复杂（参数太多）
- 如何提高网络能力
  - 局部连接
  - 权重共享
  - 汇聚操作
  - ？

大脑中的注意力

- 人脑每个时刻接收的外界输入信息非常多，包括来源于视觉、听觉、触觉的各种各样的信息。
- 但就视觉来说，眼睛每秒钟都会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。

注意力示例



注意力示例

- 当一个人在吵闹的鸡尾酒会上和朋友聊天时，尽管周围噪音干扰很多，他还是可以听到朋友的谈话内容，而忽略其他人的声音。

鸡尾酒会效应

如何实现?

- 非自主性的、基于显著性的注意力

汇聚（pooling）

- 自主性的、聚焦式注意力

会聚（focus）

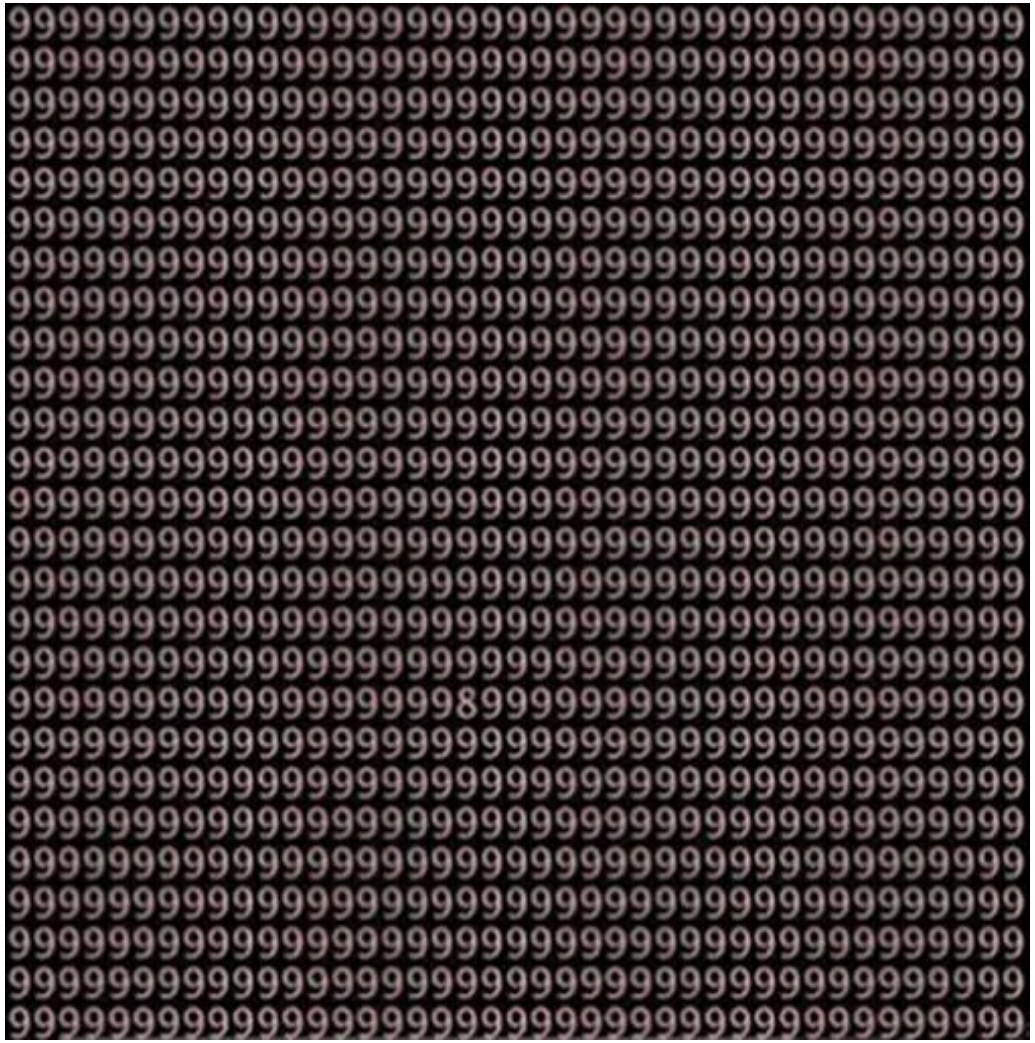
例子：阅读理解 SQuAD

The first recorded travels by Europeans to China and back date from this time. The most famous traveler of the period was the Venetian Marco Polo, whose account of his trip to "Cambaluc," the capital of the Great Khan, and of life there astounded the people of Europe. The account of his travels, Il milione (or, The Million, known in English as the Travels of Marco Polo), appeared about the year 1299. Some argue over the accuracy of Marco Polo's accounts due to the lack of mentioning the Great Wall of China, tea houses, which would have been a prominent sight since Europeans had yet to adopt a tea culture, as well the practice of foot binding by the women in capital of the Great Khan. Some suggest that Marco Polo acquired much of his knowledge through contact with Persian traders since many of the places he named were in Persian.

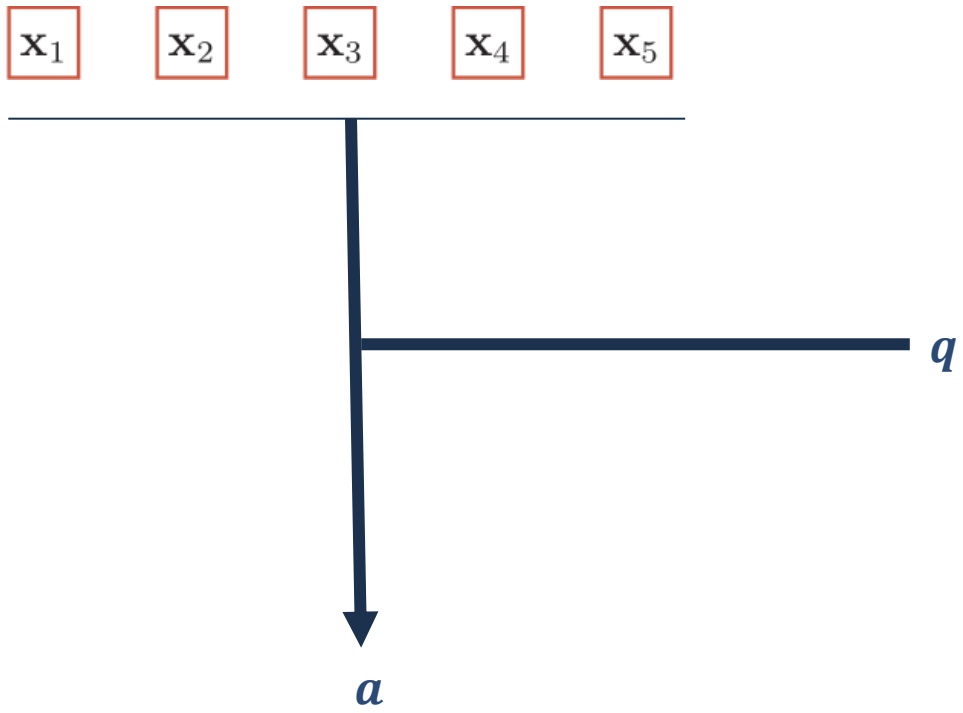
How did some suspect that Polo learned about China instead of by actually visiting it?

Answer: through contact with Persian traders

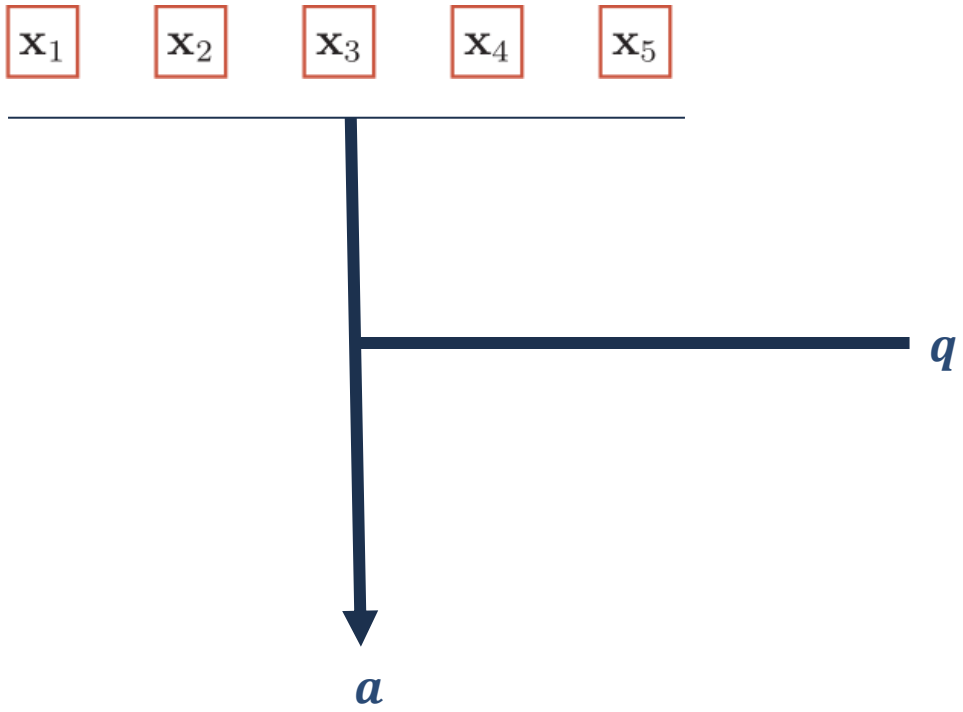
问题



Try to find 8 less than 8 seconds



问题



- 输入：
  - 用 $\boldsymbol{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N] \in \mathbb{R}^{D \times N}$ 表示 N 组输入信息，其中 D 维向量 $\boldsymbol{x}_n \in \mathbb{R}^D, n \in [1, N]$ 表示一组输入信息.
  - 从 N 个输入向量中选择出和某个特定任务相关的信息，我们需要引入一个和任务相关的表示，称为查询向量
- 输出：
  - 在给定$\boldsymbol{q}$和$\boldsymbol{X}$下，选择第n个输入向量的概率α_n

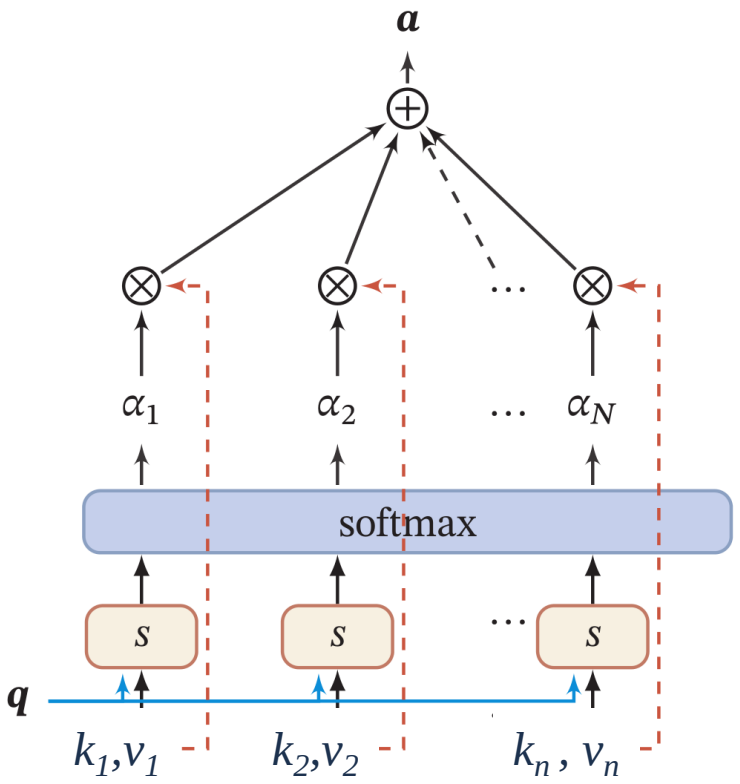
注意力层

- 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 $a_1, \ldots, a_n$； $a_i = \mathrm{s}(\mathbf{q}, \mathbf{k}_i)$
- 使用 softmax 获得注意力

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

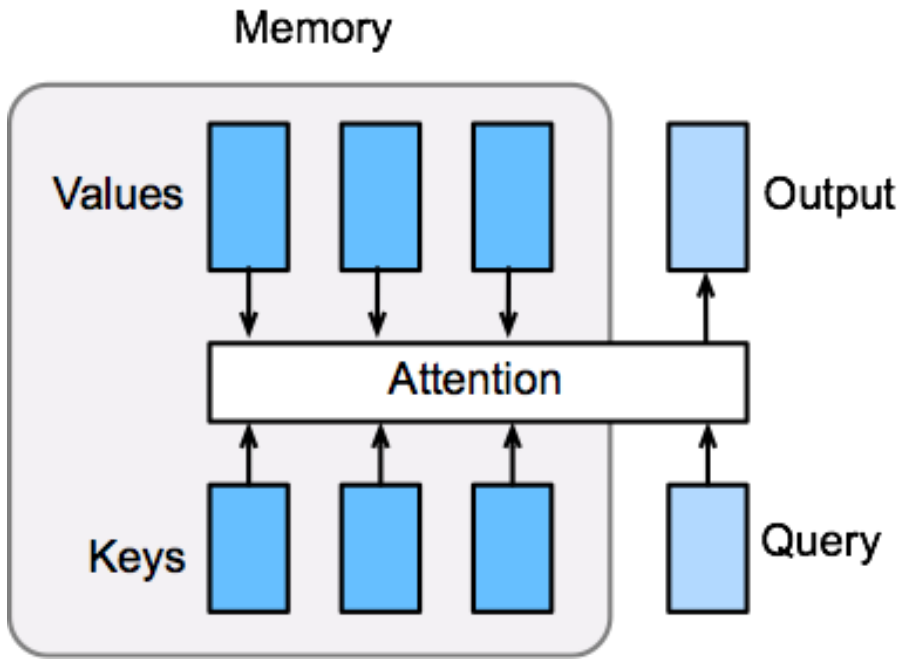
- 输出是值的加权和

$$\mathrm{output} = att((K, V), q) = \sum_{i=1}^{n} b_i \mathbf{v}_i$$



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



注意力打分函数$\alpha(q, x_n)$

- 点积模型：$s(\mathbf{q}, \mathbf{k}) = \langle \mathbf{q}, \mathbf{k} \rangle / \sqrt{d}$
- 多层感知注意力：$\mathrm{s}(\mathbf{k}, \mathbf{v}, \mathbf{q}) = \mathbf{v}^T \tanh(\mathbf{W}_k \mathbf{k} + \mathbf{W}_q \mathbf{q})$

点乘注意力

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

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

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

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

多层感知注意力(键值对注意力)

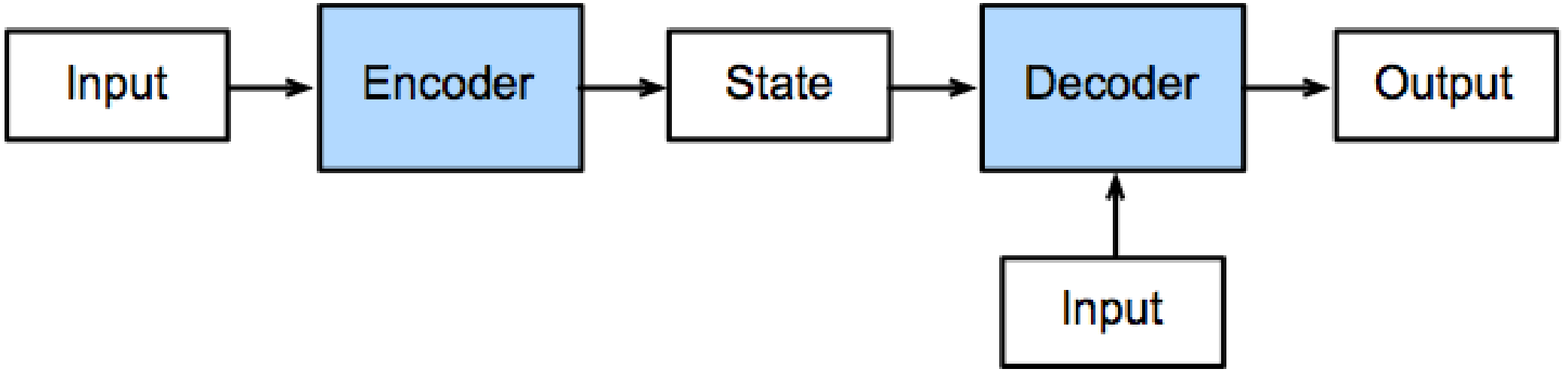
- 可学习的参数 $\mathbf{W}_k \in \mathbb{R}^{h \times d_k}, \mathbf{W}_q \in \mathbb{R}^{h \times d_q}, \mathbf{v} \in \mathbb{R}^h$

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

- 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

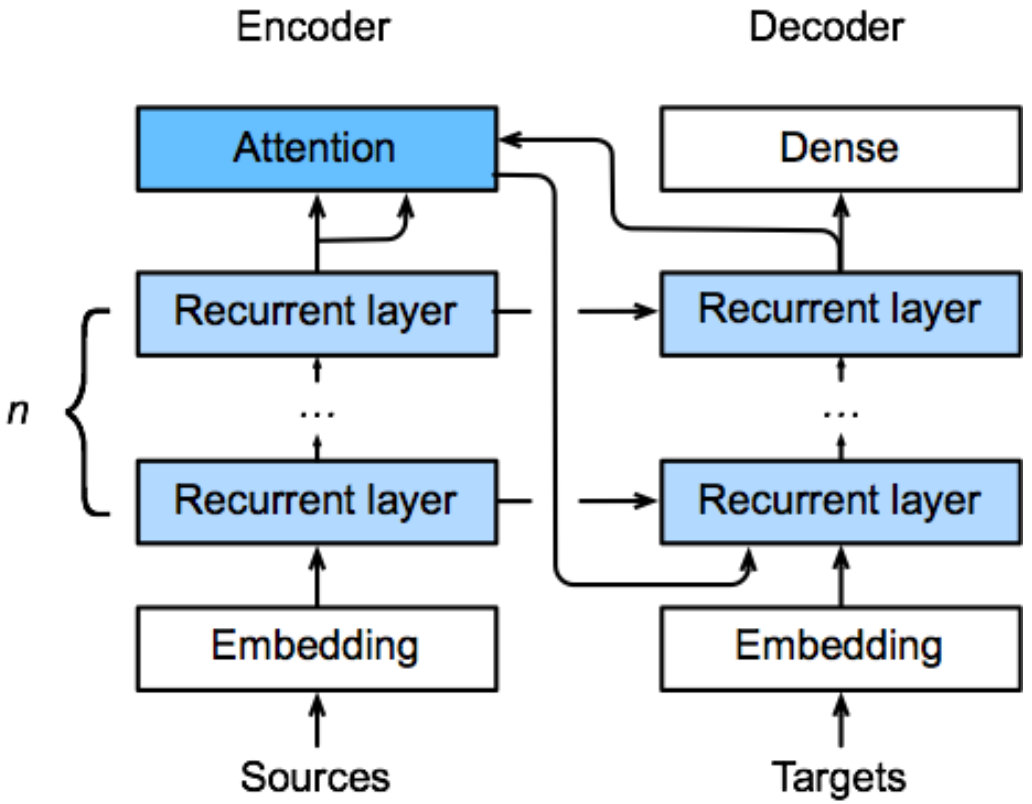
编码器 - 解码器架构

- 模型分为两部分
  - 编码器 (Encoder) 加工输入
  - 解码器 (Decoder) 生成输出



编码器 - 解码器上的注意力机制

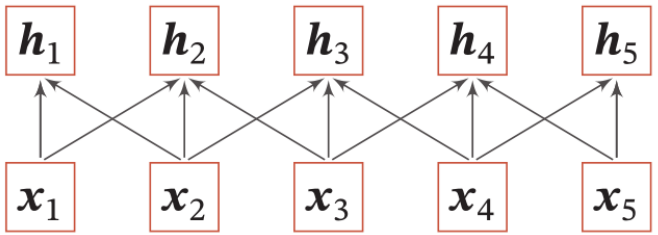
- 使用编码器中最后一个循环神经网络层的输出
- 然后，注意力输出与嵌入输出拼接，以输入解码器中的第一个循环神经网络层



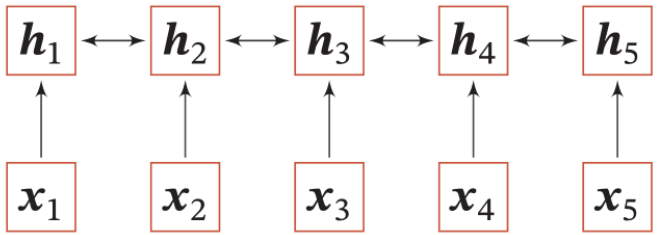
自注意力模型

自注意力模型

- 当使用神经网络来处理一个变长的向量序列时，我们通常可以使用卷积网络或循环网络进行编码来得到一个相同长度的输出向量序列。



(a) 卷积网络

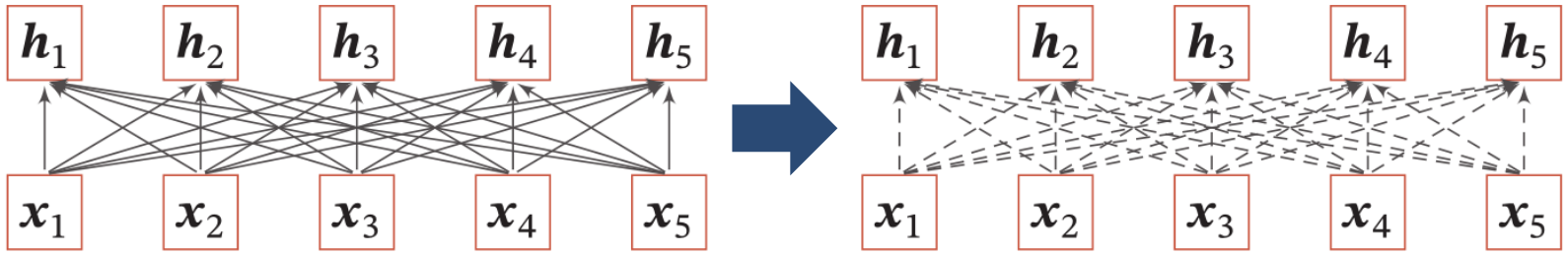


(b) 双向循环网络

只建模了输入信息的局部依赖关系

自注意力模型

- 如何建立非局部（Non-local）的依赖关系
  - 全连接?

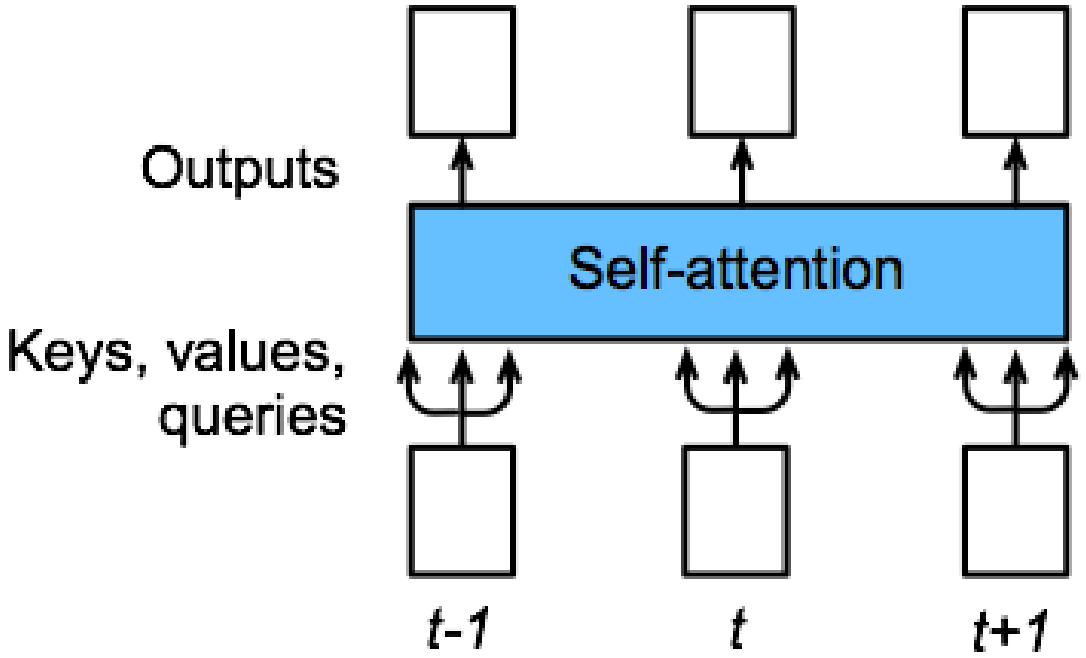


(a) 全连接模型　　(b) 自注意力模型

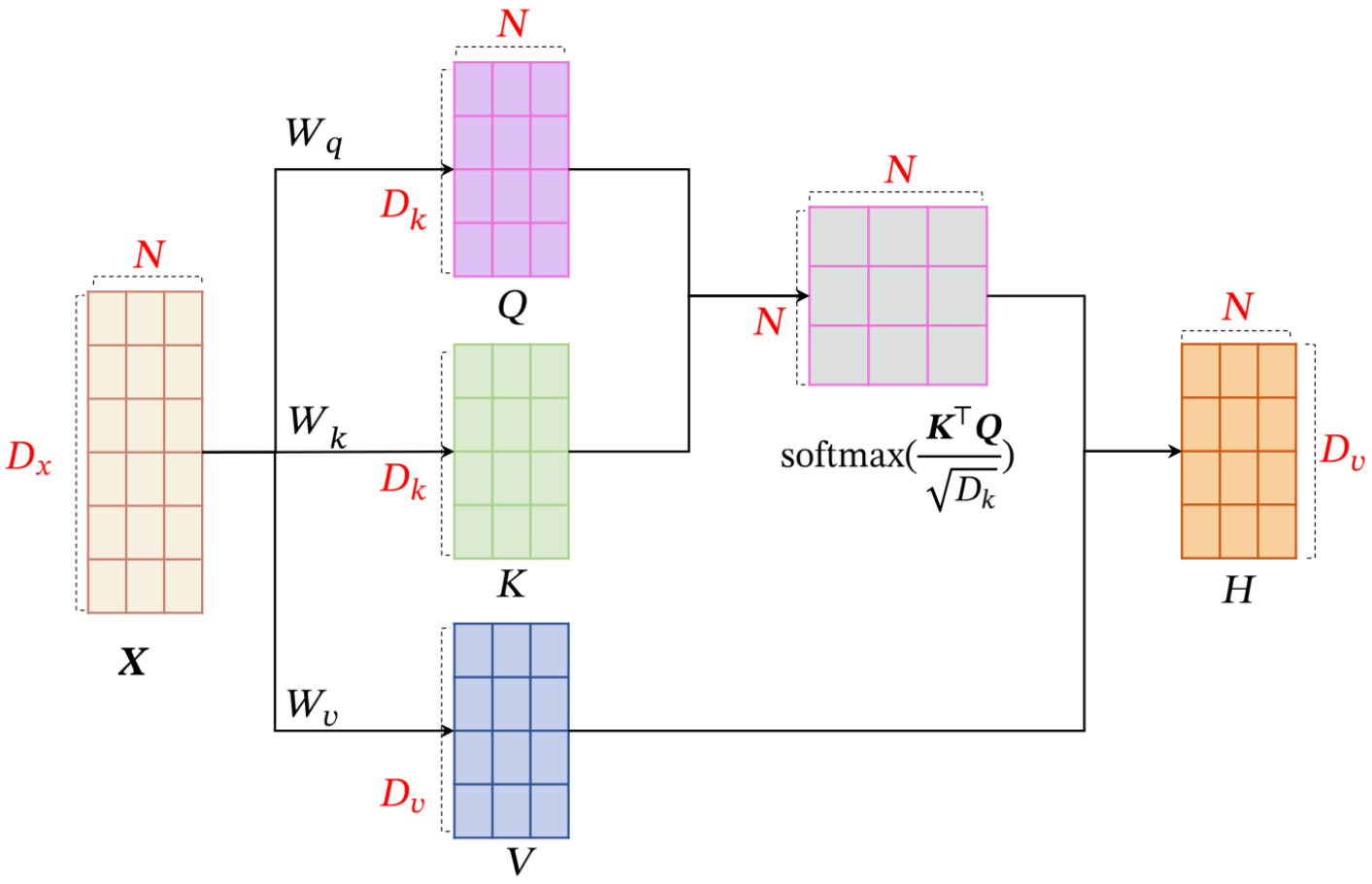
无法处理变长问题　　连接权重α_{ij} 由注意力机制动态生成

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进行编码处理，得到键（key），值（value）和查询（query）
- 并行计算
- 不保留顺序信息



QKV模式（Query-Key-Value）



自注意力模型

- 输入序列为$X = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N] \in R^{D_x \times N}$
- 首先生成三个向量序列

$$\boldsymbol{Q} = \boldsymbol{W}_q \boldsymbol{X} \in \mathbb{R}^{D_k \times N},$$
$$\boldsymbol{K} = \boldsymbol{W}_k \boldsymbol{X} \in \mathbb{R}^{D_k \times N},$$
$$\boldsymbol{V} = \boldsymbol{W}_v \boldsymbol{X} \in \mathbb{R}^{D_v \times N},$$

- 计算$\boldsymbol{h}_n$

$$\boldsymbol{h}_n = \text{att}\Big((\boldsymbol{K}, \boldsymbol{V}), \boldsymbol{q}_n\Big)$$

- 如果使用缩放点积来作为注意力打分函数，输出向量序列可以简写为

$$\boldsymbol{H} = \boldsymbol{V}\,\text{softmax}\left(\frac{\boldsymbol{K}^{\top}\boldsymbol{Q}}{\sqrt{D_k}}\right),$$

多头（multi-head）注意力

- 多头注意力（multi-head attention）
  - 利用多个查询 $\boldsymbol{Q} = [\boldsymbol{q}_1, \cdots, \boldsymbol{q}_M]$，来并行地从输入信息中选取多组信息．每个注意力关注输入信息的不同部分．

$$\text{att}\Big((\boldsymbol{K}, \boldsymbol{V}), \boldsymbol{Q}\Big) = \text{att}\Big((\boldsymbol{K}, \boldsymbol{V}), \boldsymbol{q}_1\Big) \oplus \cdots \oplus \text{att}\Big((\boldsymbol{K}, \boldsymbol{V}), \boldsymbol{q}_M\Big)$$

多头（multi-head）自注意力

- 多头自注意力（multi-head self-attention）
  - 结合了自注意力、多头注意力
  - 加入了位置编码，扩展了模型专注于不同位置的能力

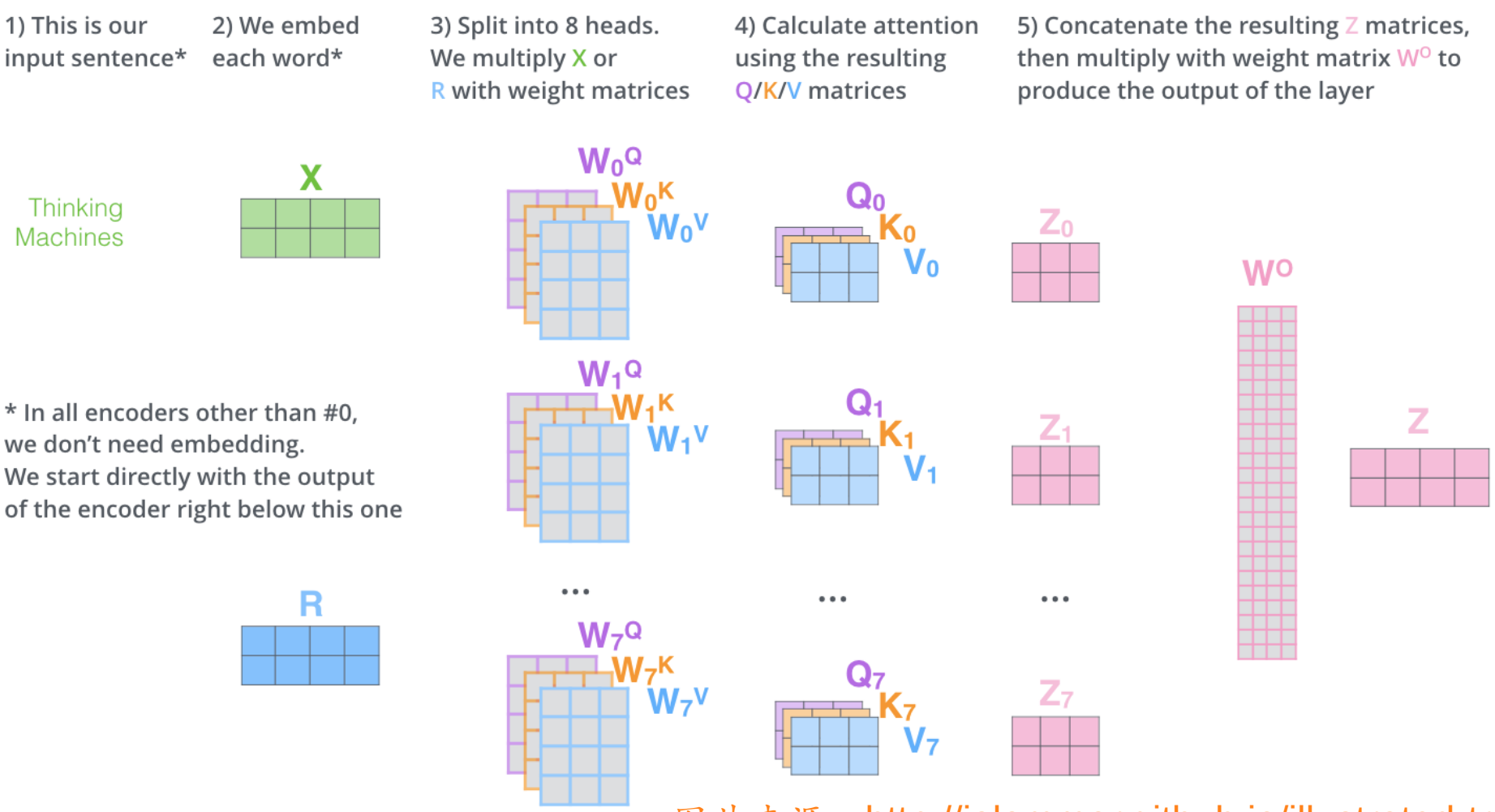
$$\text{MultiHead}(\boldsymbol{H}) = \boldsymbol{W}_o[\text{head}_1; \cdots; \text{head}_M],$$

$$\text{head}_m = \text{self-att}(\boldsymbol{Q}_m, \boldsymbol{K}_m, \boldsymbol{V}_m),$$

$$\forall m \in \{1, \cdots, M\}, \quad \boldsymbol{Q}_m = \boldsymbol{W}_q^m \boldsymbol{H}, \boldsymbol{K} = \boldsymbol{W}_k^m \boldsymbol{H}, \boldsymbol{V} = \boldsymbol{W}_v^m \boldsymbol{H},$$

其中 $\boldsymbol{W}_o \in \mathbb{R}^{D_h \times M d_v}$ 为输出投影矩阵，$\boldsymbol{W}_q^m \in \mathbb{R}^{D_k \times D_h}$，$\boldsymbol{W}_k^m \in \mathbb{R}^{D_k \times D_h}$，$\boldsymbol{W}_v^m \in \mathbb{R}^{D_v \times D_h}$ 为投影矩阵，$m \in \{1, \cdots, M\}$.

多头（multi-head）自注意力模型



图片来源：http://jalammar.github.io/illustrated-transformer/

序列生成模型

序列数据的潜在规律

- 以自然语言为例，

面包上涂黄油

面包上涂袜子

- 后一个句子在人脑的语义整合时需要更多的处理时间，更不符合自然语言规则。

- 规则是概率：自然语言理解 → 一个句子中单词排序的可能性/合理性

序列概率模型

- 给定一个序列样本，其概率为

$$\mathrm{p}(x_{1:\mathrm{T}}) = \mathrm{p}(x_1, x_2, \cdots, x_{\mathrm{T}})$$

- 和一般的概率模型类似，序列概率模型有两个基本问题：
  - （1）学习问题：给定一组序列数据，估计这些数据背后的概率分布；
  - （2）生成问题：从已知的序列分布中生成新的序列样本。

序列概率模型

- 给定一个序列样本，其概率为

$$\mathrm{p}(x_{1:\mathrm{T}}) = \mathrm{p}(x_1, x_2, \cdots, x_{\mathrm{T}})$$

- 序列数据有两个特点：
  - （1）样本是变长的；
  - （2）样本空间为非常大。
- 对于一个长度为T的序列，其样本空间为$|\mathrm{V}|^T$。因此，我们很难用已知的概率模型来直接建模整个序列的概率。

序列概率模型

- 序列概率

$$\mathrm{p}(x_{1:\mathrm{T}}) = \prod_{\mathrm{t}} \mathrm{p}(x_{\mathrm{t}}|x_{1:\mathrm{t}-1})$$
$$\approx \prod_{\mathrm{t}} \mathrm{p}(x_{\mathrm{t}}|x_{\mathrm{t}-1}, \cdots, x_{\mathrm{t}-n+1}) = \prod_{\mathrm{t}} g(h_t)$$

- 因此，序列数据的概率密度估计问题可以转换为单变量的条件概率估计问题，即给定$x_{1:\mathrm{t}-1}$时x_{t}的条件概率$\mathrm{p}(x_{\mathrm{t}}|x_{1:\mathrm{t}-1})$。

序列概率模型

- 序列概率
  - $\mathrm{p}(x_1, x_2, \cdots, x_\mathrm{T})$
  - $= \prod_\mathrm{t} \mathrm{p}(x_\mathrm{t} | x_{\mathrm{t}-1}, \cdots, x_1)$
  - $\approx \prod_\mathrm{t} \mathrm{p}(x_\mathrm{t} | x_{\mathrm{t}-1}, \cdots, x_{\mathrm{t}-n+1}) = \prod_\mathrm{t} g(h_t)$

- 因此，序列数据的概率密度估计问题可以转换为单变量的条件概率估计问题，即给定$x_{1:(t-1)}$ 时x_t的条件概率$p(x_t | x_{1:(t-1)})$。

自回归生成模型

给定N个序列数据$\{\mathbf{x}_{1:T_n}^{(n)}\}_{n=1}^{N}$，序列概率模型需要学习一个模型$p_\theta(x|\mathbf{x}_{1:(t-1)})$来最大化整个数据集的对数似然函数。

$$\max_\theta \sum_{n=1}^{N} \log p_\theta(\mathbf{x}_{1:T_n}^{(n)}) = \max_\theta \sum_{n=1}^{N} \sum_{t=1}^{T_n} \log p_\theta(x_t^{(n)}|\mathbf{x}_{1:(t-1)}^{(n)}). \quad (15.5)$$

- 在这种序列模型方式中，每一步都需要将前面的输出作为当前步的输入，是一种自回归（autoregressive）的方式。
  - 自回归生成模型（Autoregressive Generative Model）

序列生成

- 自回归生成模型（Autoregressive Generative Model）

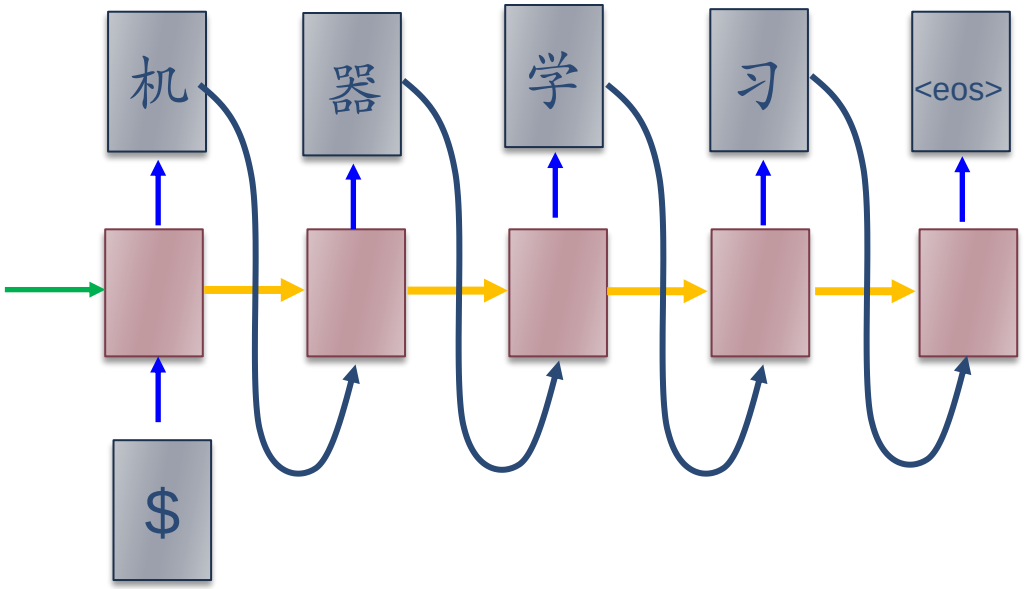
一旦通过最大似然估计训练了模型$p_\theta(x|\mathbf{x}_{1:(t-1)})$，就可以通过时间顺序来生成一个完整的序列样本。令$\hat{x}_t$为在第$t$时根据分布$p_\theta(x|\hat{\mathbf{x}}_{1:(t-1)})$生成的词，

$$\hat{x}_t \sim p_\theta(x|\hat{\mathbf{x}}_{1:(t-1)}), \tag{15.6}$$

其中$\hat{\mathbf{x}}_{1:(t-1)} = \hat{x}_1, \cdots, \hat{x}_{t-1}$为前面$t-1$步中生成的前缀序列。

- 非自回归生成模型
  - 同时生成所有词

序列生成



自回归的方式可以生成一个无限长度的序列。
为了避免这种情况，通常会设置一个特殊的符号“<eos>”来表示序列的结束。
在训练时，每个序列样本的结尾都加上符号“<eos>”。在测试时，一旦生成了符号“<eos>”，就中止生成过程。

生成最可能序列

- 当使用自回归模型生成一个最可能的序列时，生成过程是一种从左到右的贪婪式搜索过程。在每一步都生成最可能的词。
- 这种贪婪式的搜索方式是次优的，生成的序列并不保证是全局最优的。

$$\prod_{t=1}^{T} \max_{x_t \in \mathcal{V}} p_\theta(x_t | \hat{\mathbf{x}}_{1:(t-1)}) \leq \max_{\mathbf{x}_{1:T} \in \mathcal{V}^T} \prod_{t=1}^{T} p_\theta(x | \mathbf{x}_{1:(t-1)}).$$

贪婪搜索

- 我们在预测期间在 seq2seq 模型中使用了贪婪搜索
- 它可能不是最理想的

贪婪搜索:
$0.5\times0.4\times0.4\times0.6=0.048$

时间步	1	2	3	4
A	0.5	0.1	0.2	0.0
B	0.2	0.4	0.2	0.2
C	0.2	0.3	0.4	0.2
<eos>	0.1	0.2	0.2	0.6

更好的选择:
$0.5\times0.3\times0.6\times0.6=0.054$

时间步	1	2	3	4
A	0.5	0.1	0.1	0.1
B	0.2	0.4	0.6	0.2
C	0.2	0.3	0.2	0.1
<eos>	0.1	0.2	0.1	0.6

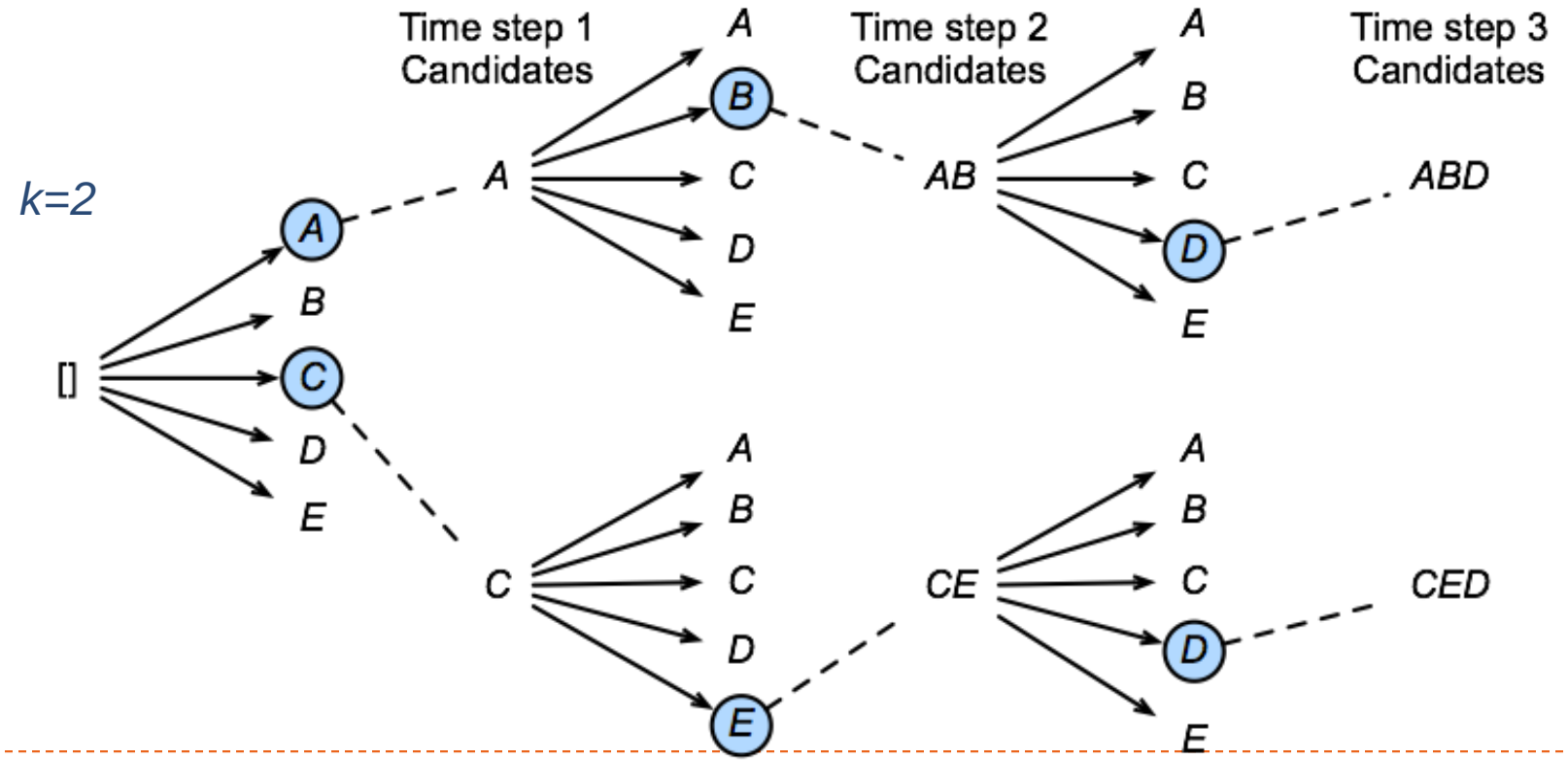
穷举搜索（exhaustive search）

- 对于每个可能的序列，计算其概率并选择最佳序列
- 如果输出词汇量大小为 n，并且最大序列长度为 T，那么我们需要检查 n^T 序列
  - 这在计算上是不可行的

$$n = 10000, T = 10: n^T = 10^{40}$$

束搜索

- 每次都保留最好的 k（束搜索）候选
- 通过向候选束添加新项目，来搜索 kn 序列，然后保留前k 个



束搜索

- 时间复杂度为 O(knT)

$$k = 5, n = 10000, T = 10:$$
$$knT = 5 \times 10^5$$

- 每个候选的最终得分是

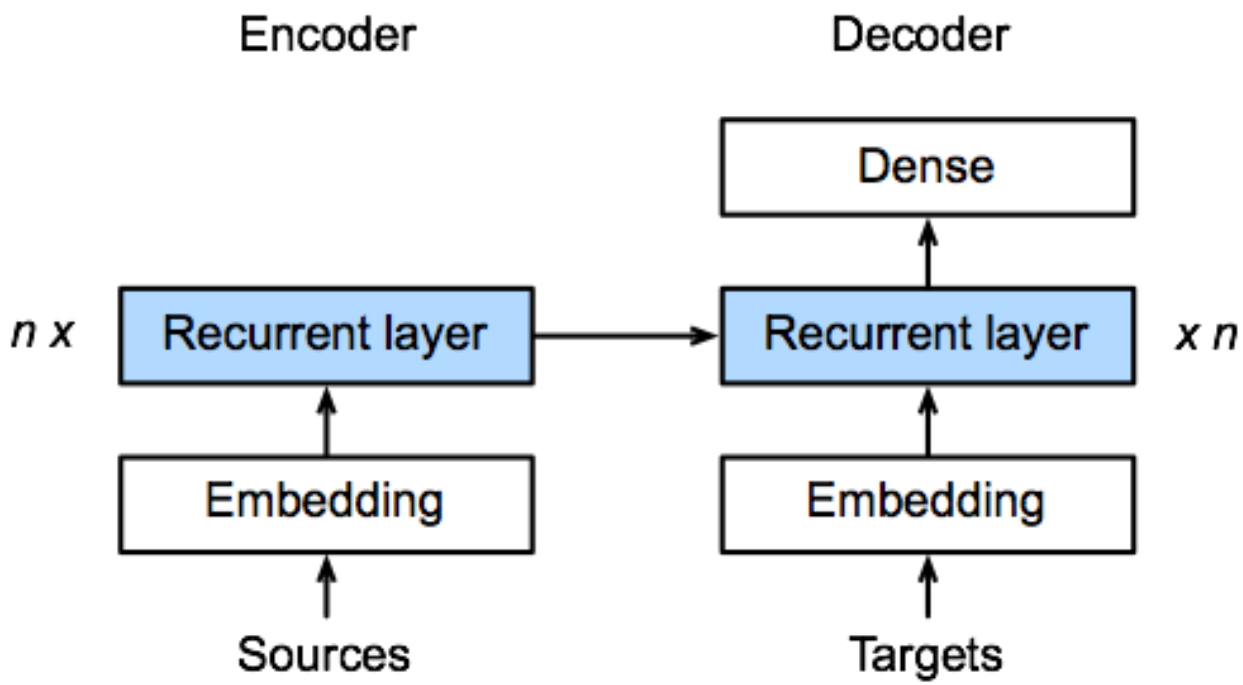
$$\frac{1}{L^\alpha}\log\mathbb{P}(y_1, \cdots, y_L) = \frac{1}{L^\alpha}\sum_{t'=1}^{L}\log\mathbb{P}(y_{t'} \mid y_1, \cdots, y_{t'-1}, \boldsymbol{c})$$

  - 常用值$\alpha = 0.75$

序列到序列模型

Seq2seq 模型

- 编码器是没有输出层的标准 RNN 模型
- 编码器在上一时间步骤中的隐含状态用作解码器的初始隐藏状态



Seq2seq 模型

序列到序列模型的目标是估计条件概率

$$p_\theta(\mathbf{y}_{1:T}|\mathbf{x}_{1:S}) = \prod_{t=1}^{T} p_\theta(y_t|\mathbf{y}_{1:(t-1)}, \mathbf{x}_{1:S}), \tag{15.96}$$

其中 $\mathbf{y}_t \in \mathcal{V}$ 为词表 $\mathcal{V}$ 中的某个词。

给定一组训练数据 $\{(\mathbf{x}_{S_n}, \mathbf{y}_{T_n})\}_{n=1}^{N}$，我们可以使用最大似然估计来训练模型参数。

$$\max_\theta \sum_{n=1}^{N} \log p_\theta(\mathbf{y}_{1:T_n}|\mathbf{x}_{1:S_n}). \tag{15.97}$$

一旦训练完成，模型就可以根据一个输入序列 $\mathbf{x}$ 来生成最可能的目标序列，

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}), \tag{15.98}$$

具体的生成过程可以通过贪婪方法或束搜索来完成。

Seq2seq 模型

编码器 首先使用一个循环神经网络 f_{enc} 来编码输入序列 $\boldsymbol{x}_{1:S}$ 得到一个固定维数的向量 $\boldsymbol{u}$，$\boldsymbol{u}$ 一般为编码循环神经网络最后时刻的隐状态.

$$\boldsymbol{h}_t^{\text{enc}} = f_{\text{enc}}(\boldsymbol{h}_{t-1}^{\text{enc}}, \boldsymbol{e}_{x_{t-1}}, \theta_{\text{enc}}), \qquad \forall t \in [1:S],$$

$$\boldsymbol{u} = \boldsymbol{h}_S^{\text{enc}},$$

其中 $f_{\text{enc}}(\cdot)$ 为编码循环神经网络，可以为 LSTM 或 GRU，其参数为 θ_{enc}，$\boldsymbol{e}_x$ 为词 x 的词向量.

Seq2seq 模型

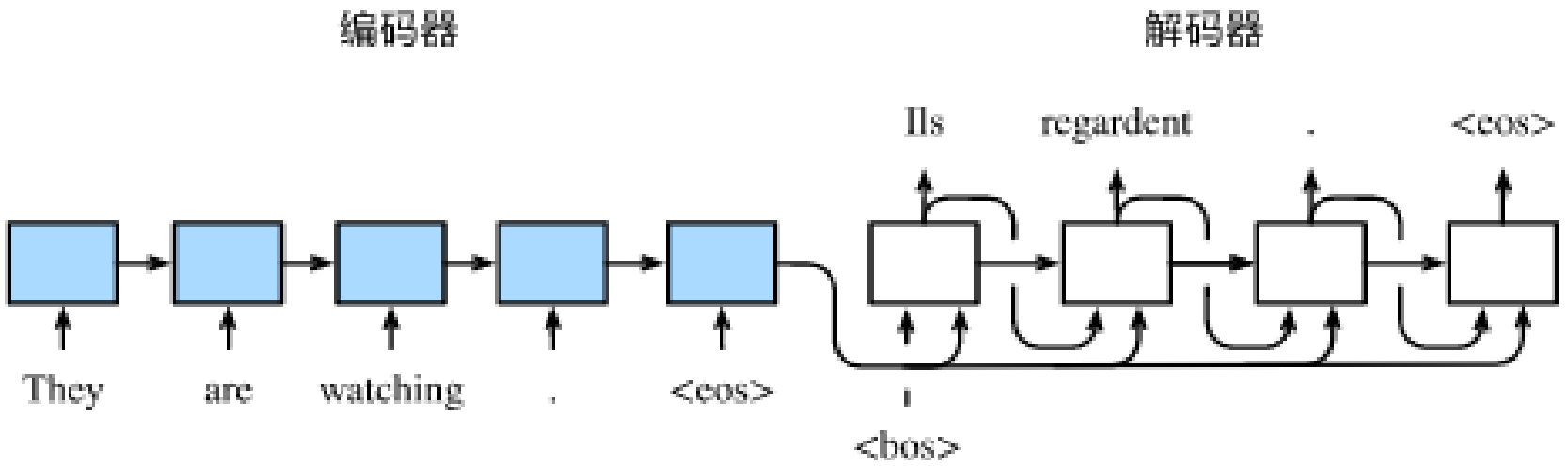
解码器 在生成目标序列时，使用另外一个循环神经网络 f_{dec} 来进行解码. 在解码过程的第 t 步时，已生成前缀序列为 $\boldsymbol{y}_{1:(t-1)}$. 令 $\boldsymbol{h}_t^{\text{dec}}$ 表示在网络 f_{dec} 的隐状态，$\boldsymbol{o}_t \in (0,1)^{|\mathcal{V}|}$ 为词表中所有词的后验概率，则

$$\begin{aligned}
\boldsymbol{h}_0^{\text{dec}} &= \boldsymbol{u}, \\
\boldsymbol{h}_t^{\text{dec}} &= f_{\text{dec}}(\boldsymbol{h}_{t-1}^{\text{dec}}, \boldsymbol{e}_{y_{t-1}}, \theta_{\text{dec}}), \\
\boldsymbol{o}_t &= g(\boldsymbol{h}_t^{\text{dec}}, \theta_o),
\end{aligned}$$

其中 $f_{\text{dec}}(\cdot)$ 为解码循环神经网络，$g(\cdot)$ 为最后一层为 Softmax 函数的前馈神经网络，θ_{dec} 和 θ_o 为网络参数，$\boldsymbol{e}_y$ 为 y 的词向量，y_0 为一个特殊符号，比如 $\langle EOS \rangle$.

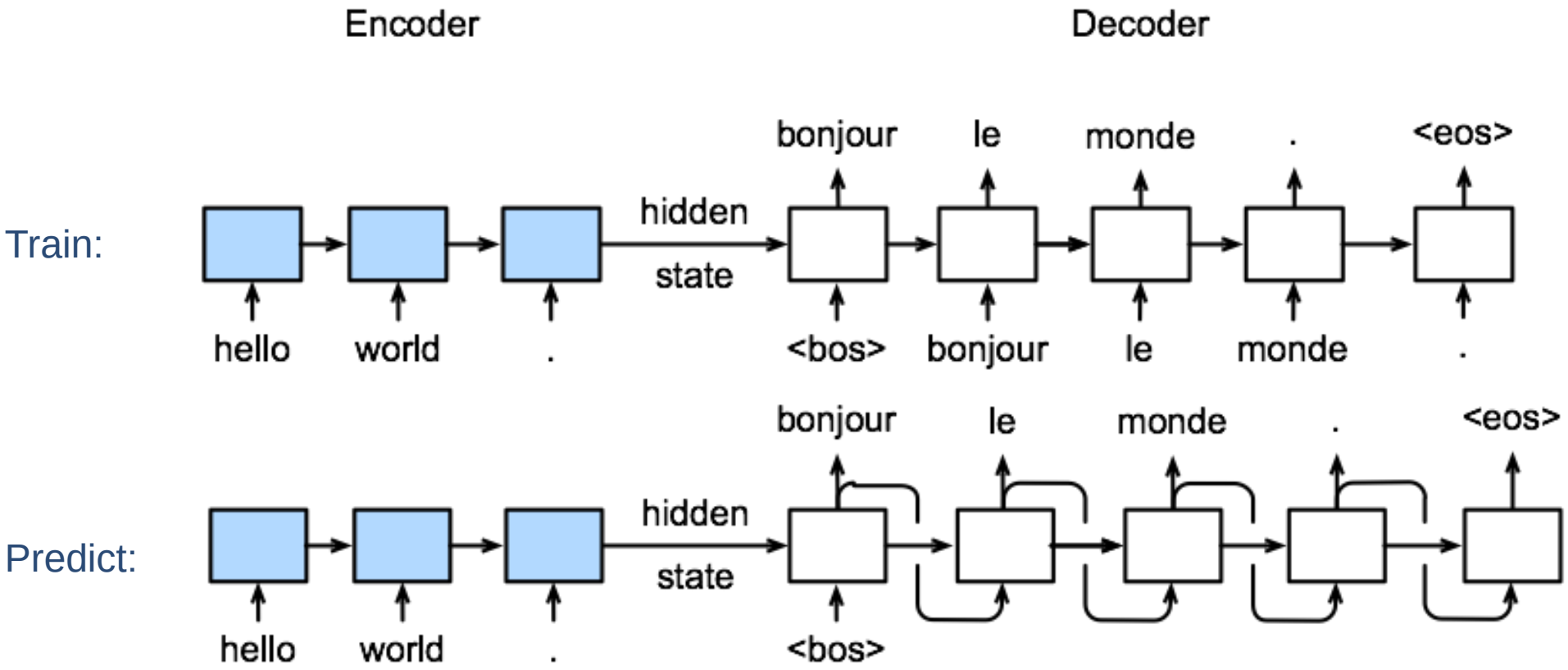
Seq2seq 模型

- 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



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



评价方法

困惑度

- 困惑度（Perplexity）是信息论的一个概念，可以用来衡量一个分布的不确定性。
- 给定一个测试文本集合，一个好的序列生成模型应该使得测试集合中的句子的联合概率尽可能高。

- 困惑度可以衡量模型分布与样本经验分布之间的契合程度。困惑度越低则两个分布越接近。

困惑度

假设测试集合共有独立同分布的 N 个序列 $\{\mathbf{x}_{1:T_n}^{(n)}\}_{n=1}^{N}$。我们可以用模型 $p_\theta(\mathbf{x})$ 对每个序列计算其概率 $p_\theta(\mathbf{x}_{1:T_n}^{(n)})$，整个测试集的联合概率为

$$\prod_{n=1}^{N} p_\theta\left(\mathbf{x}_{1:T_n}^{(n)}\right) = \prod_{n=1}^{N}\prod_{t=1}^{T_n} p_\theta\left(x_t^{(n)}|\mathbf{x}_{1:(t-1)}^{(n)}\right). \tag{15.35}$$

模型 $p_\theta(\mathbf{x})$ 的困惑度定义为

$$\begin{aligned}
\mathrm{PPL}(\theta) &= 2^{-\frac{1}{T}\sum_{n=1}^{N}\log p_\theta\left(\mathbf{x}_{1:T_n}^{(n)}\right)} && (15.36)\\
&= 2^{-\frac{1}{T}\sum_{n=1}^{N}\sum_{t=1}^{T_n}\log p_\theta\left(x_t^{(n)}|\mathbf{x}_{1:(t-1)}^{(n)}\right)} && (15.37)\\
&= \left(\prod_{n=1}^{N}\prod_{t=1}^{T_n} p_\theta\left(x_t^{(n)}|\mathbf{x}_{1:(t-1)}^{(n)}\right)\right)^{-1/T}, && (15.38)
\end{aligned}$$

其中 $T = \sum_{n=1}^{N} T_n$ 为测试数据集中序列的总长度。可以看出，困惑度为每个词条件概率 $p_\theta\left(x_t^{(n)}|\mathbf{x}_{1:(t-1)}^{(n)}\right)$ 的几何平均数的倒数。测试集中所有序列的概率越大，困惑度越小，模型越好。

BLEU

- BLEU（Bilingual Evaluation Understudy）是衡量模型生成序列和参考序列之间的N元词组（N-Gram）的重合度，最早用来评价机器翻译模型的质量，目前也广泛应用在各种序列生成任务中。

BLEU

假设从模型分布 p_θ 中生成一个候选（Candidate）序列 $\mathbf{x}$，从真实数据分布中采样出的一组参考（Reference）序列 $\mathbf{s}^{(1)}, \cdots, \mathbf{s}^{(K)}$，我们首先从生成序列中提取 N 元组合的集合 $\mathcal{W}$，并计算 N 元组合的精度（Precision），

$$P_n(\mathbf{x}) = \frac{\sum\limits_{w \in \mathcal{W}} \min\left(c_w(\mathbf{x}), \max\limits_{k=1}^{K} c_w(\mathbf{s}^{(k)})\right)}{\sum\limits_{w \in \mathcal{W}} c_w(\mathbf{x})}, \tag{15.39}$$

其中 $c_w(\mathbf{x})$ 是 N 元组合 w 在生成序列 $\mathbf{x}$ 中出现的次数，$c_w(\mathbf{s}^{(k)})$ 是 N 元组合 w 在参考序列 $\mathbf{s}^{(k)}$ 中出现的次数。N 元组合的精度 $P_n(\mathbf{x})$ 是计算生成序列中的 N 元组合有多少比例在参考序列中出现。

BLEU 是通过计算不同长度的 N 元组合的精度，并进行几何加权平均而得到。

$$\text{BLEU-N}(\mathbf{x}) = b(\mathbf{x}) \times \exp\Big(\sum_{n=1}^{N} w_n \log P_n\Big), \tag{15.41}$$

$$b(\mathbf{x}) = \begin{cases} 1 & \text{if} \quad l_x > l_s \\ \exp\left(1 - l_s/l_x\right) & \text{if} \quad l_x \leq l_s \end{cases}$$

ROUGE

- ROUGE（Recall-Oriented Understudy for Gisting Evaluation）最早应用于文本摘要领域。和BLEU类似，但ROUGE计算的是召回率（Recall）。

假设从模型分布 p_θ 中生成一个候选（Candidate）序列 $\mathbf{x}$，从真实数据分布中采样出的一组参考（Reference）序列 $\mathbf{s}^{(1)}, \cdots, \mathbf{s}^{(K)}$，令 $\mathcal{W}$ 为从参考序列中提取N元组合的集合，ROUGE-N的定义为

$$\text{ROUGE-N}(\mathbf{x}) = \frac{\sum_{k=1}^{K} \sum_{w \in \mathcal{W}} \min\left(c_w(\mathbf{x}), c_w\left(\mathbf{s}^{(k)}\right)\right)}{\sum_{k=1}^{K} \sum_{w \in \mathcal{W}} c_w\left(\mathbf{s}^{(k)}\right)}, \tag{15.42}$$

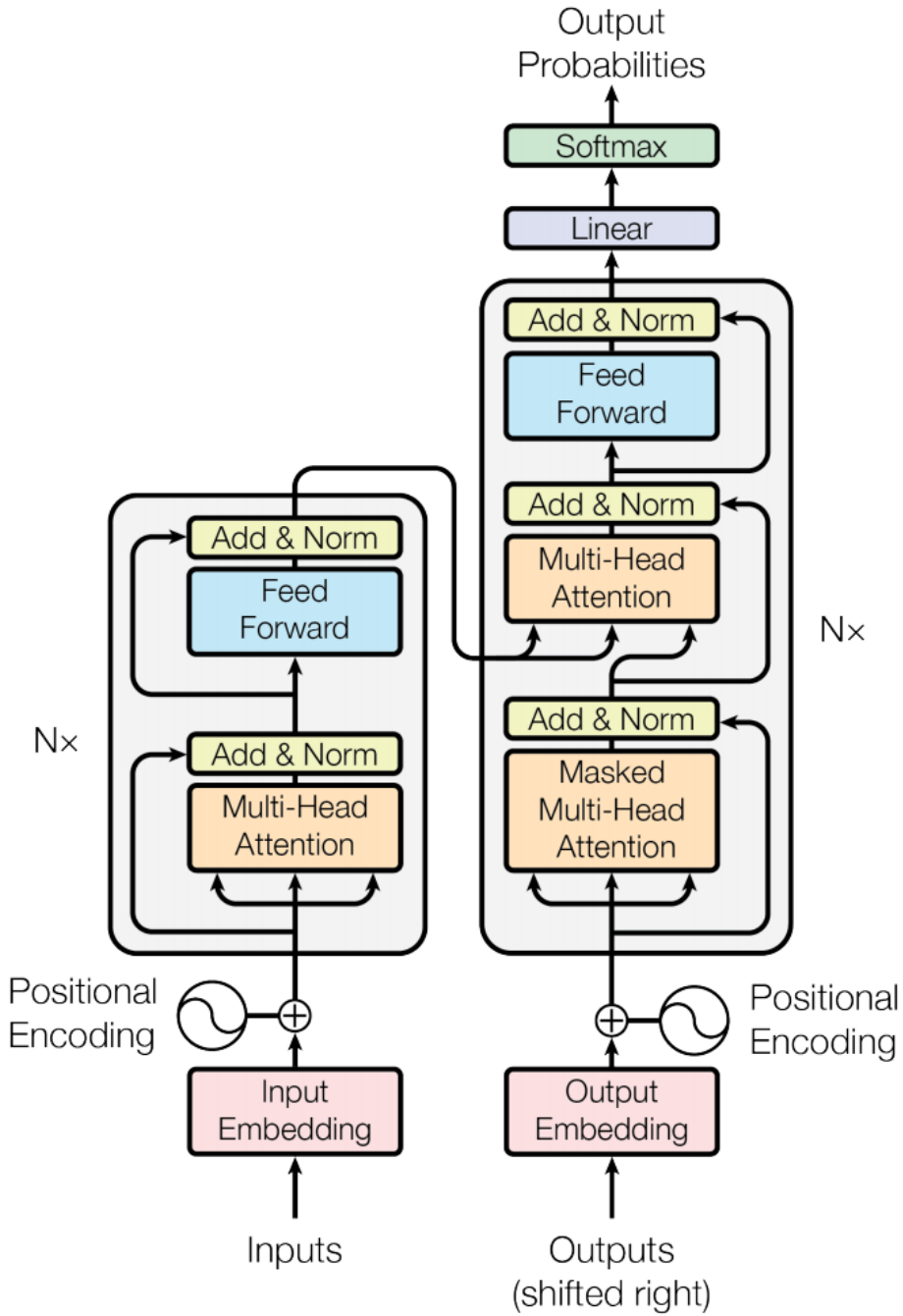
其中 $c_w(\mathbf{x})$ 是N元组合 w 在生成序列 $\mathbf{x}$ 中出现的次数，$c_w(\mathbf{s}^{(k)})$ 是N元组合 w 在参考序列 $\mathbf{s}^{(k)}$ 中出现的次数。

Transformer模型

Attention Is All You Need

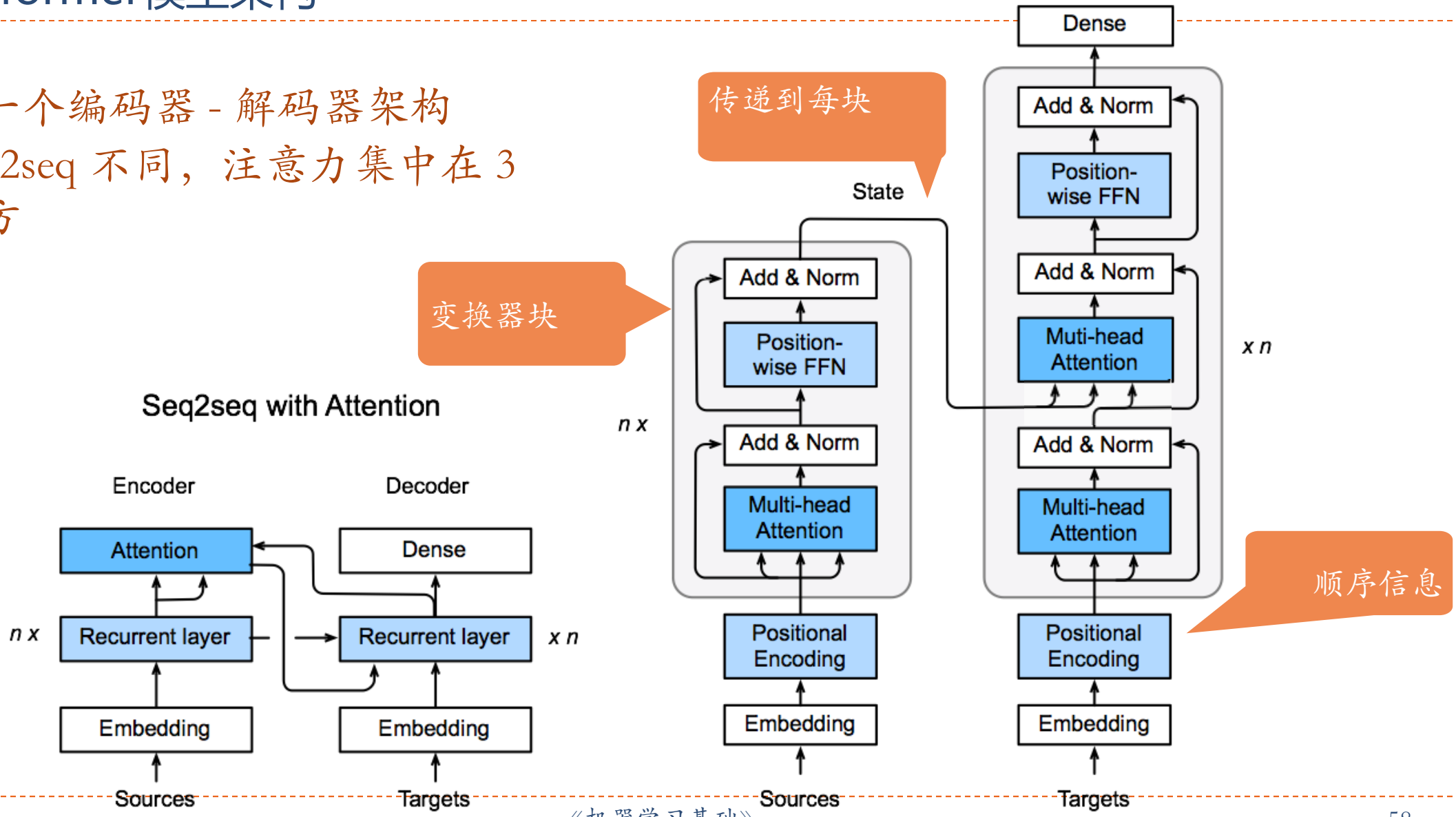
Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

Transformer模型架构

- 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

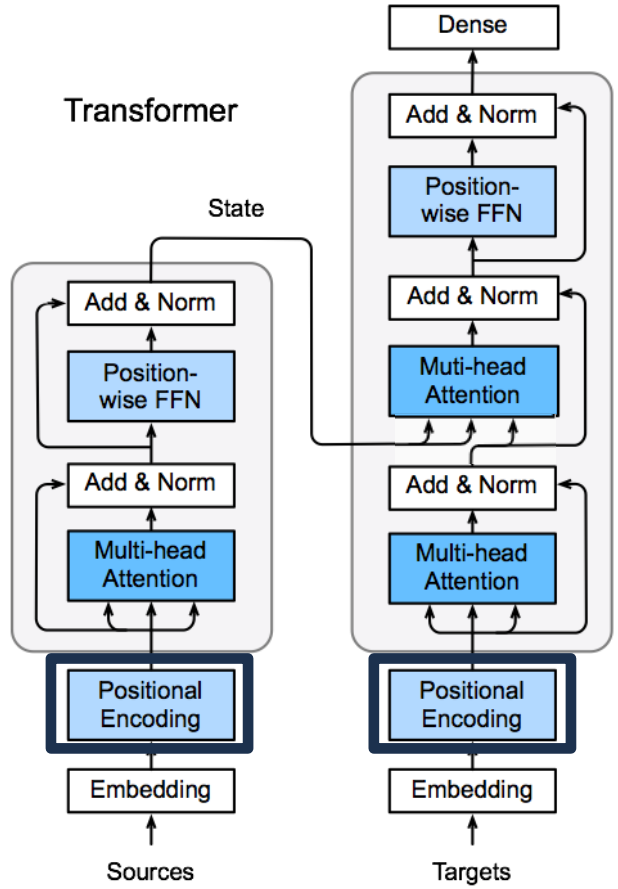


位置编码

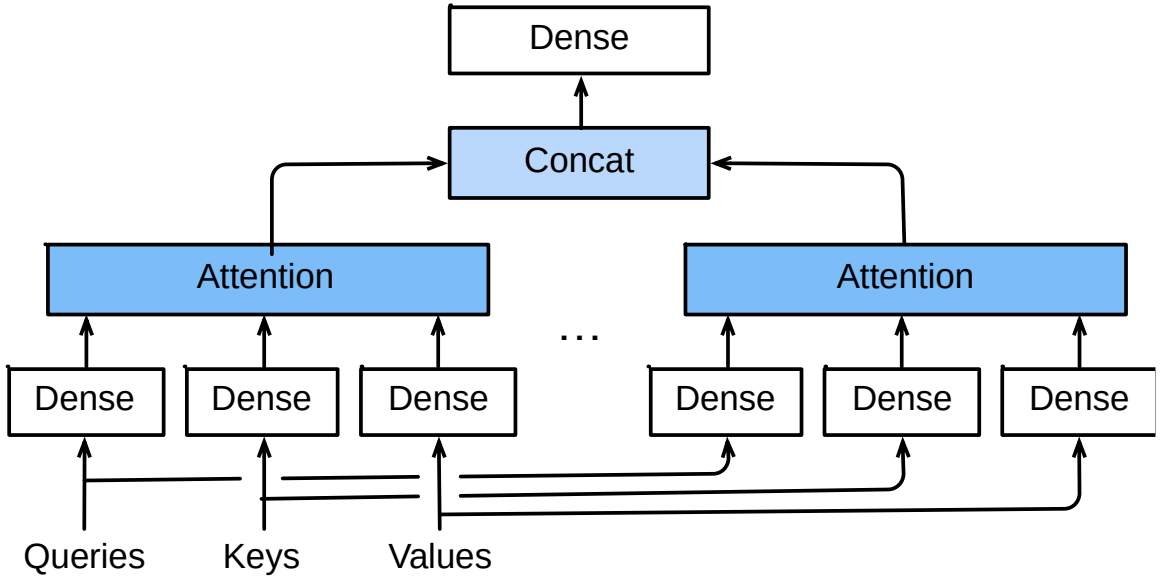
- 假设嵌入 $X \in \mathbb{R}^{l \times d}$ 输出的形状（序列长度，嵌入维度）
- 创建 $P \in \mathbb{R}^{l \times d}$
- 输出 $X + P$

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

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



多头注意力机制（Multi-head Self-attention）

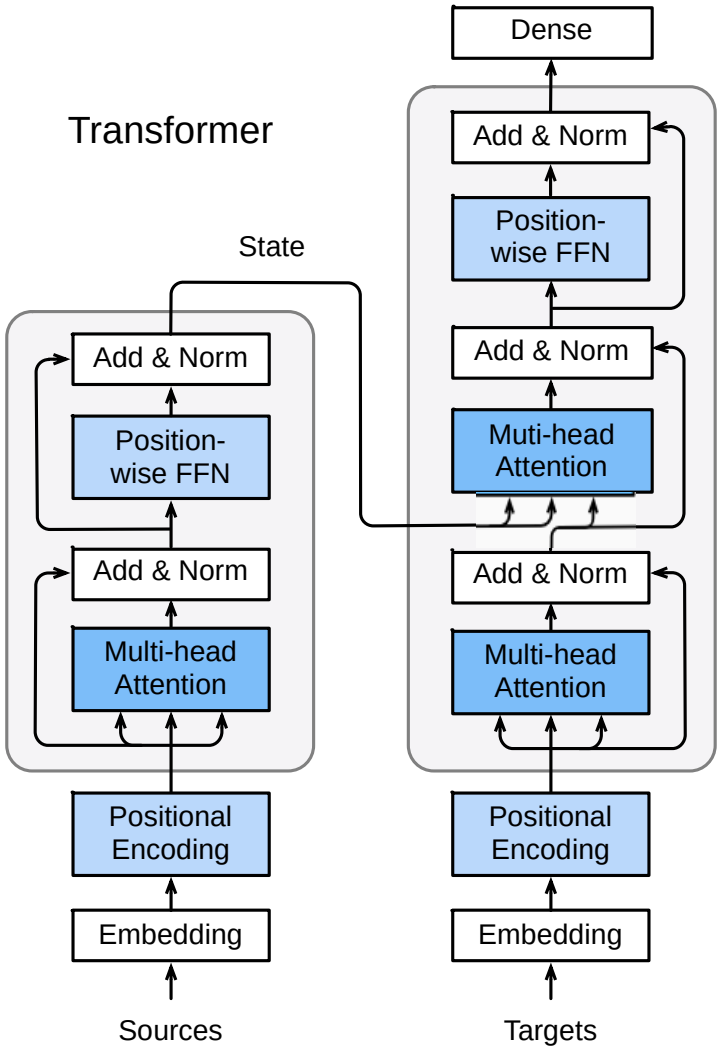


$$\mathbf{W}_q^{(i)} \in \mathbb{R}^{p_q \times d_q}, \mathbf{W}_k^{(i)} \in \mathbb{R}^{p_k \times d_k}, \text{ and } \mathbf{W}_v^{(i)} \in \mathbb{R}^{p_v \times d_v}$$

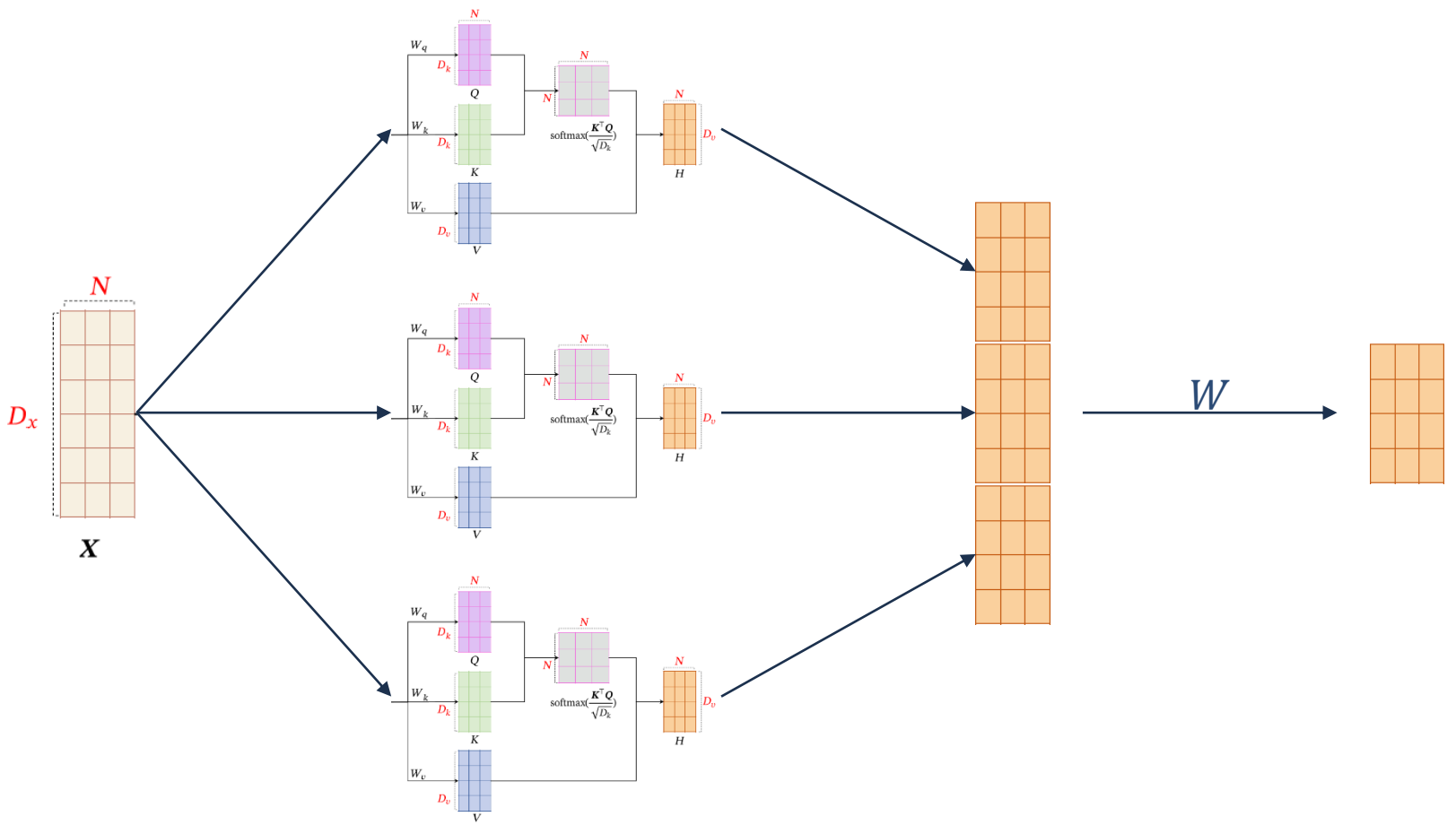
$$\mathbf{o}^{(i)} = \text{attention}(\mathbf{W}_q^{(i)}\mathbf{q}, \mathbf{W}_k^{(i)}\mathbf{k}, \mathbf{W}_v^{(i)}\mathbf{v})$$

for $i = 1, \ldots, h$

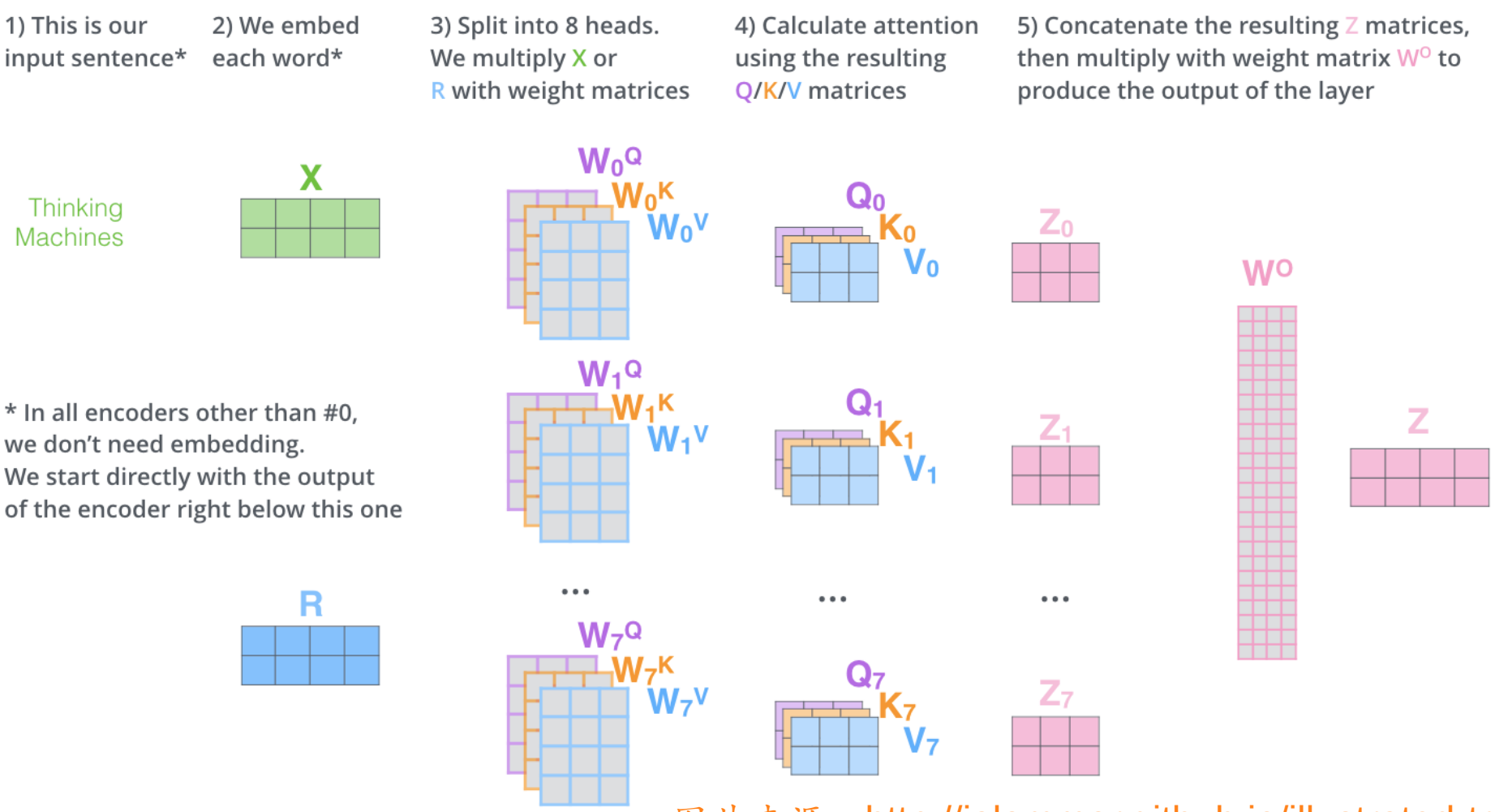
$$\mathbf{o} = \mathbf{W}_o \begin{bmatrix} \mathbf{o}^{(1)} \\ \vdots \\ \mathbf{o}^{(h)} \end{bmatrix}$$



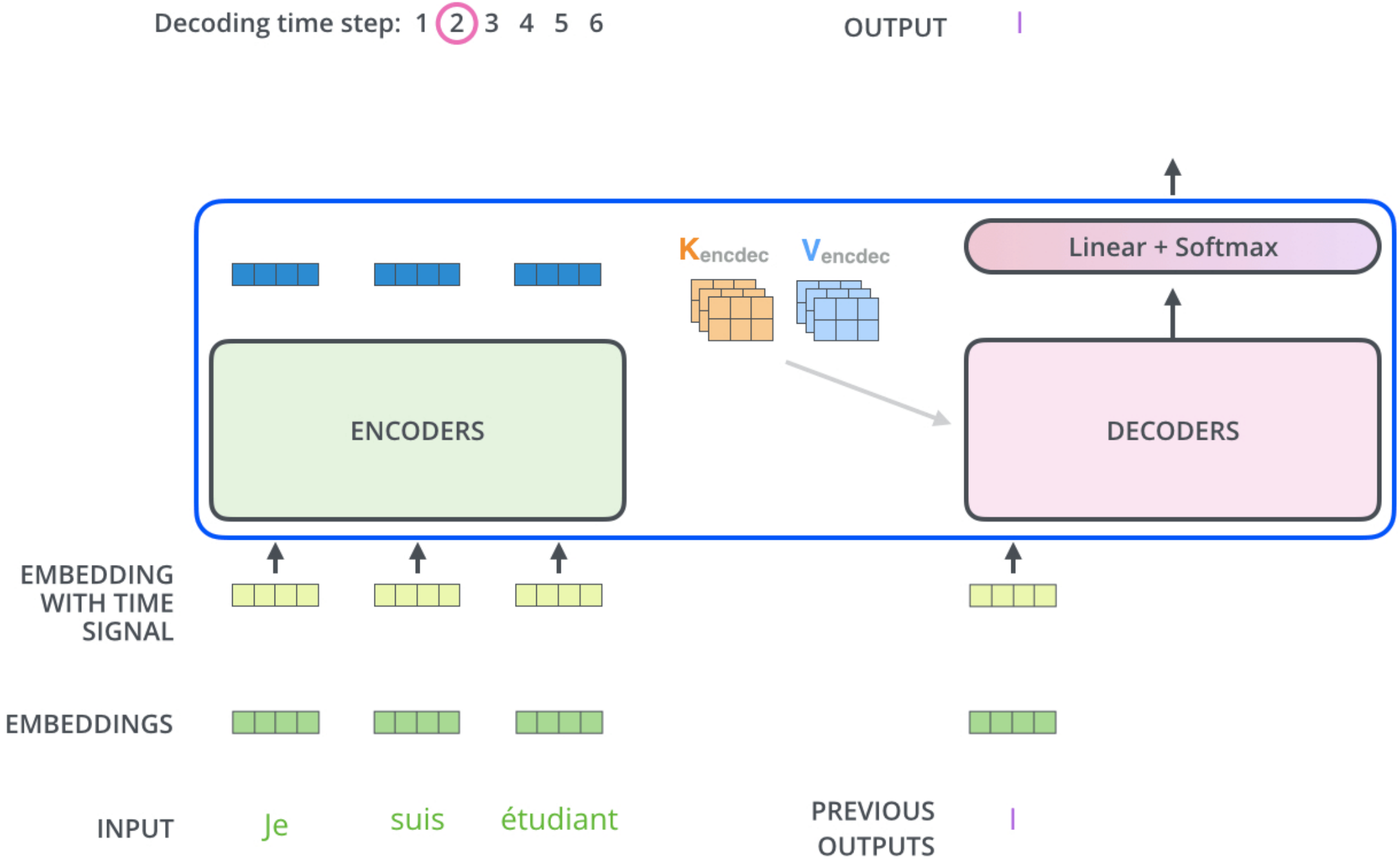
多头（multi-head）自注意力模型



多头（multi-head）自注意力模型

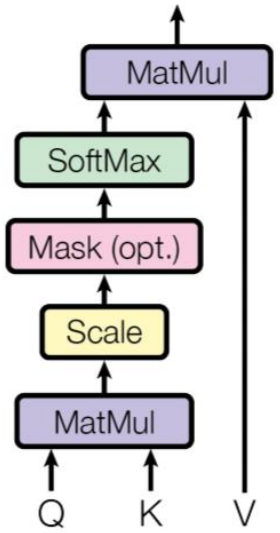


图片来源：http://jalammar.github.io/illustrated-transformer/

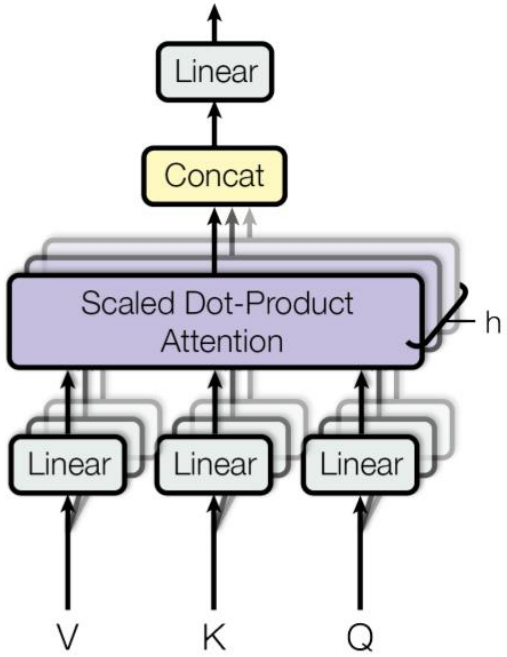

Decoding time step: 1 2 3 4 5 6
OUTPUT
I
Linear + Softmax
K encdec
V encdec
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant
PREVIOUS OUTPUTS
I

Transformer

Scaled Dot-Product Attention



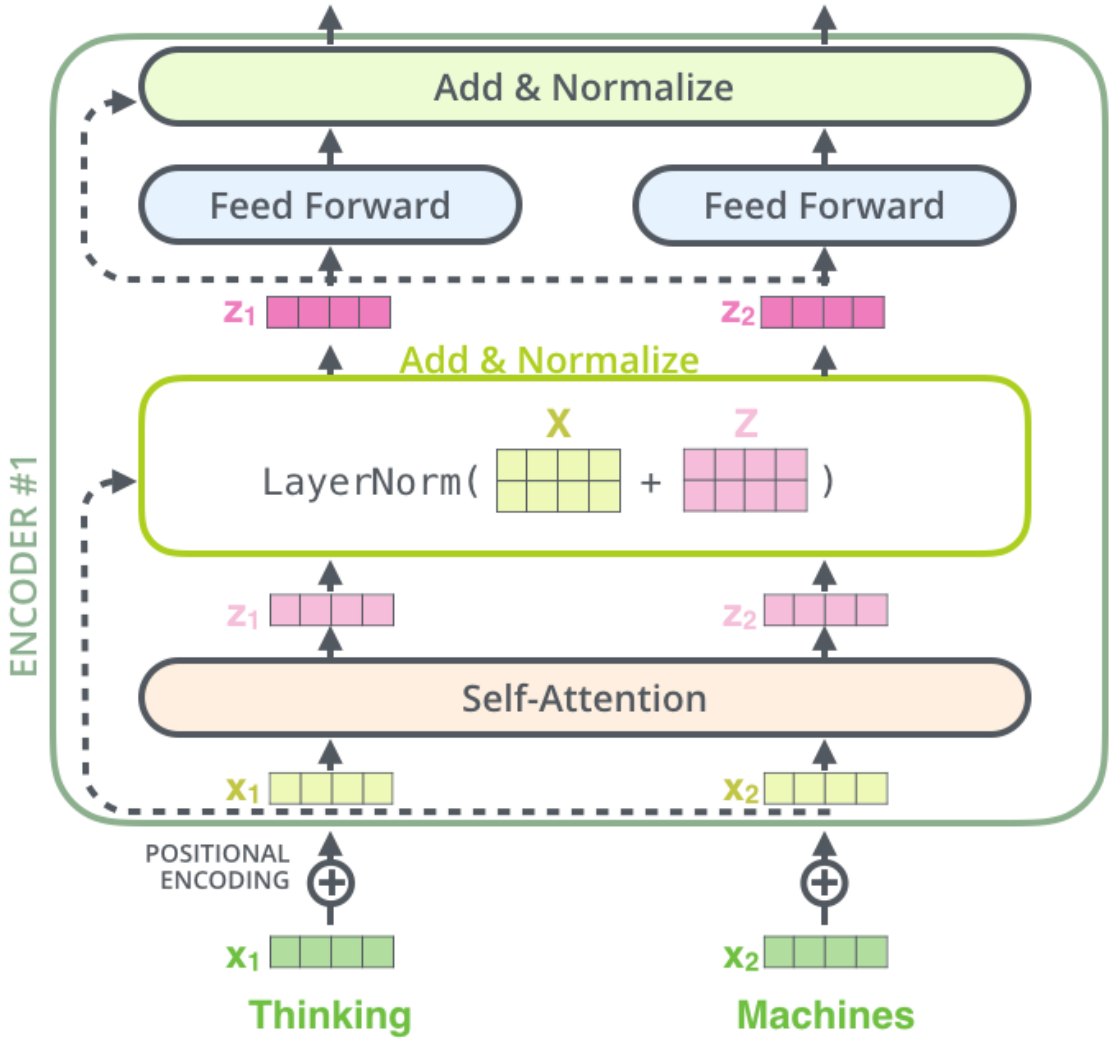
Multi-Head Attention



Transformer Encoder

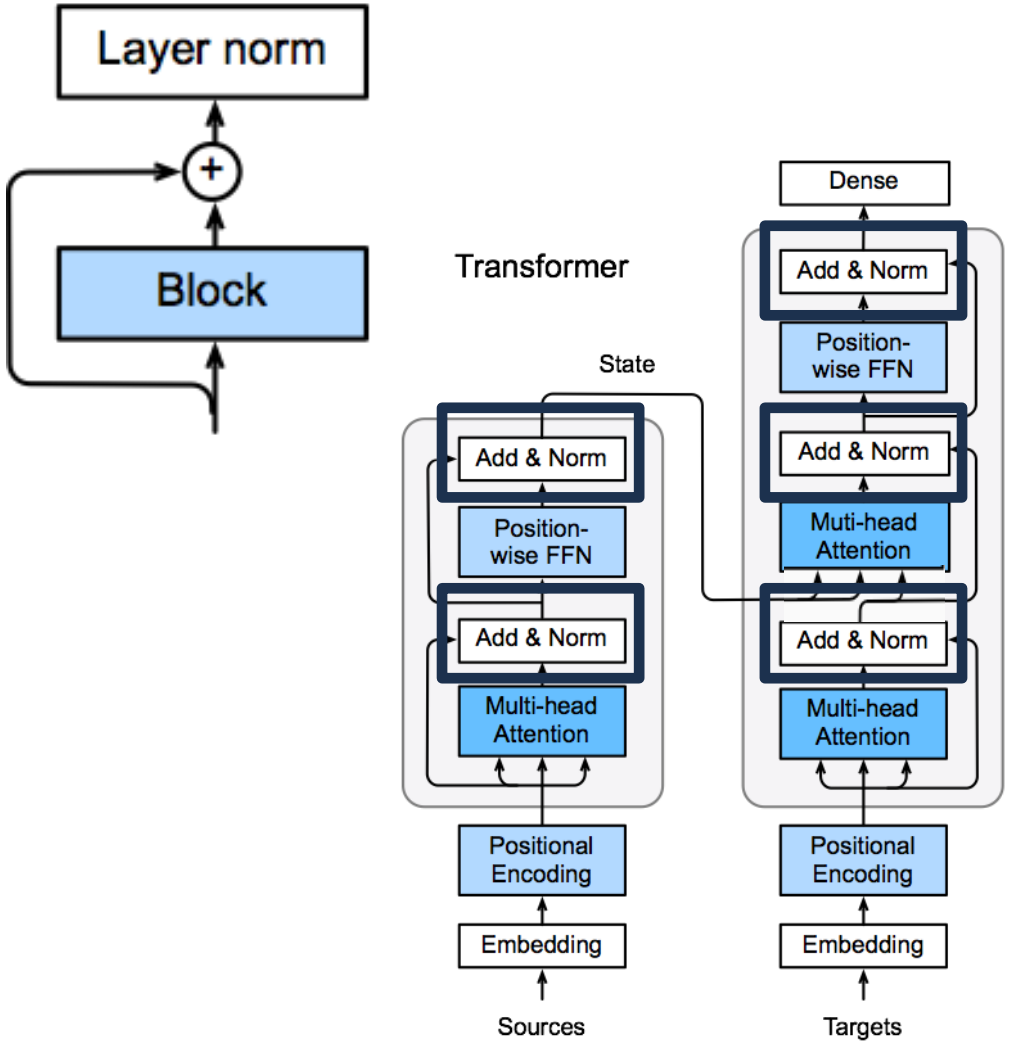
图片来源：http://jalammar.github.io/illustrated-transformer/

- 仅仅自注意力还不够
- 其它操作
  - 位置编码
  - 层归一化
  - 直连边
  - 逐位的FNN

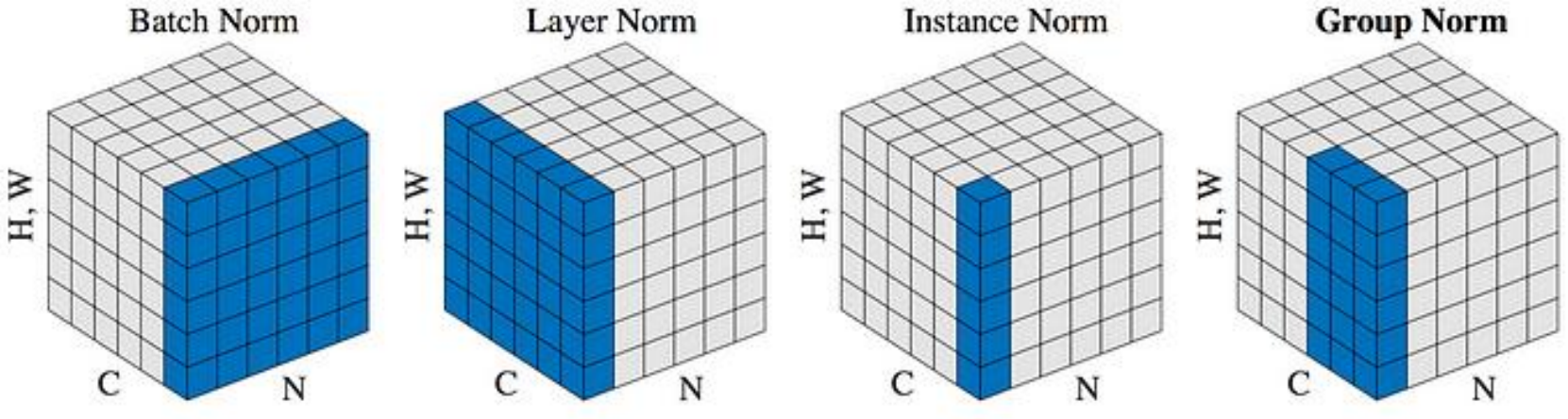


添加与归一化

- 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）

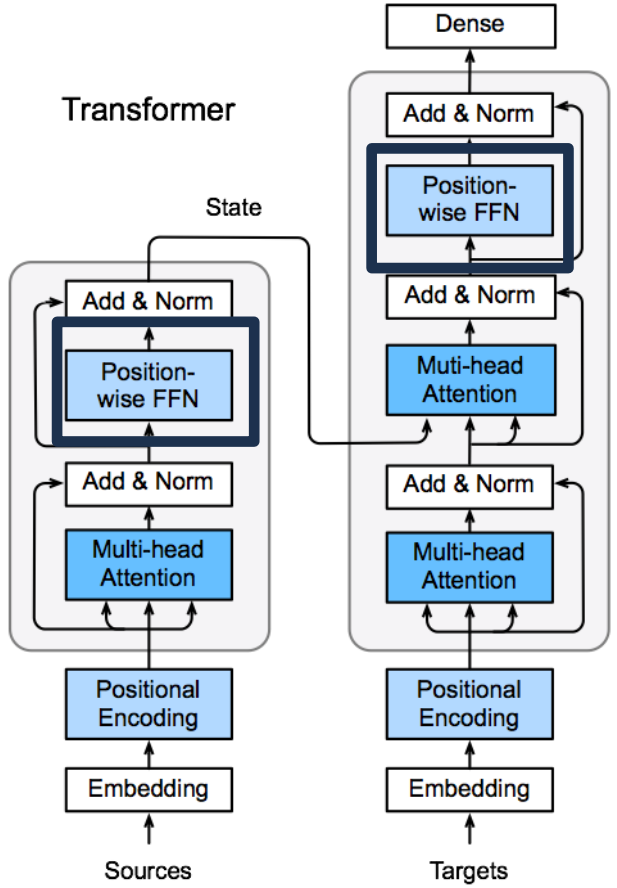


更多的归一化



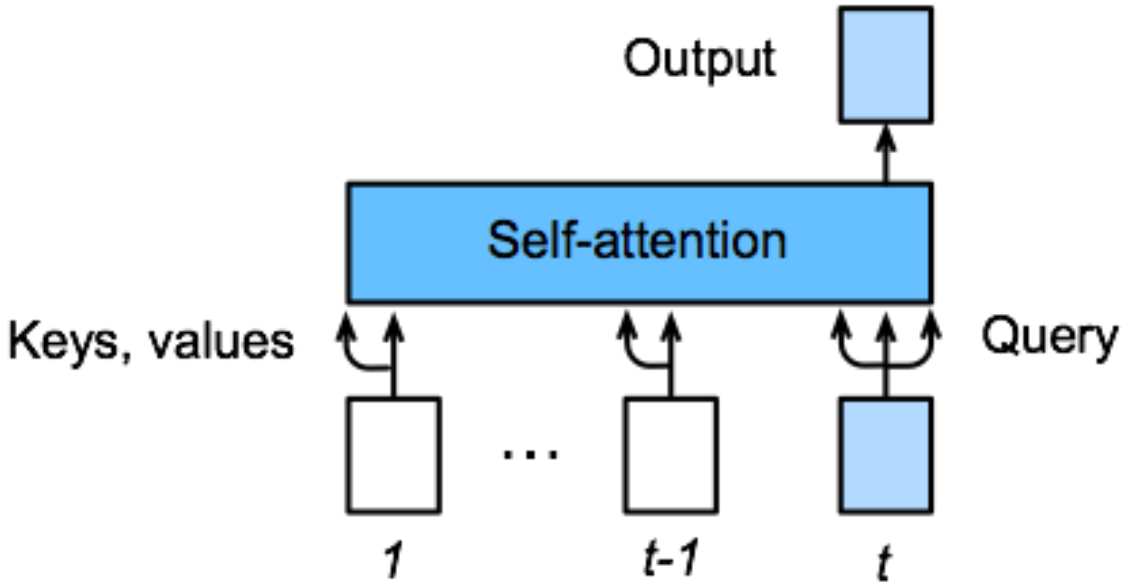
位置前馈网络

- 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



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



复杂度分析

模型	每层复杂度	序列操作数	最大路径长度
CNN	$O(kLd^2)$	$O(1)$	$O(\log_k(L))$
RNN	$O(Ld^2)$	$O(L)$	$O(L)$
Transformer	$O(L^2d)$	$O(1)$	$O(1)$

k卷积核大小　L序列长度　d维度