序列生成模型

- **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；
- **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

根据概率的乘法公式，序列$\mathbf{x}_{1:T}$的概率可以写为

$$\mathrm{p}(x_{1:T}) = \prod_{t=1}^{T} p(x_t|x_{1:t-1})$$

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

$$\max_\theta \sum_{n=1}^{N} log p_\theta(x_{1:T_n}^{(n)}) = \max_\theta \sum_{n=1}^{N} \sum_{t=1}^{T_n} log p_\theta(x_t^{(n)}|x_{1:(t-1)}^{(n)})$$

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。因此这一类模型也称为自回归生成模型（Autoregressive Generative Model）：N 元统计模型何深度序列模型。

序列生成

一旦通过最大似然估计训练了模型$p_\theta(x|x_{1:t-1})$，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$p_\theta(x|\hat{x}_{1:t-1})$生成的词，

束搜索　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，

$$\hat{x}_t = arg\max_{x \in V} p_\theta(x|\hat{x}_{1:t-1})$$

这种贪婪式的搜索方式是次优的，生成的序列$\hat{x}_{1:T}$并不保证是全局最优的。

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

束搜索（Beam Search）：在每一步的生成中，生成 K 个最可能的前缀序列。

N 元统计模型

一元模型　当 n=1 时，序列$\mathrm{x}_{1:T}$中每个词都和其它词独立，和它的上下文无关。每个位置上的词都是从多项分布独立生成的。在多项分布中，$\theta = [\theta_1, \cdots, \theta_{|V|}]$为词表中每个词被抽取的概率。

$$p(\mathrm{x}_{1:T}|\theta) = \prod_{t=1}^{T} p(x_t) = \prod_{k=1}^{|V|} \theta_k^{m_k}$$

给定一组训练集$\{x^{(n)_{1:T_n}}\}_{n=1}^{N'}$，其对数似然函数为：

$$\log\prod_{t=1}^{N'}p(x_{1:T_n}^{(n)}|\theta)=log\prod_{k=1}^{|V|}\theta_k^{m_k}=\sum_{k=1}^{|V|}m_k log\theta_k$$

$$\max_{\theta}\sum_{k=1}^{|V|}m_k log\theta_k\,,s.t.\sum_{k=1}^{|V|}\theta_k=1$$

$$\mathrm{L}(\theta,\lambda)=\sum_{k=1}^{|V|}m_k log\theta_k+\lambda(\sum_{k=1}^{|V|}\theta_k-1)$$

$$\lambda=-\sum_{v=1}^{V}m_v\,,\theta_k^{ML}=\frac{m_k}{\sum_{k=1}^{|V|}m_k}=\frac{m_k}{\bar{m}}$$

N 元模型　N 元模型中的条件概率$\mathrm{p}(\mathrm{x}_t|\mathrm{x}_{(t-n+1):(t-1)})$也可以通过最大似然函数来得到。

$$p\left(x_t|x_{(t-n+1):(t-1)}\right)=\frac{m(x_{(t-n+1):t})}{m(x_{(t-n+1):(t-1)})}$$

平滑技术　N 元模型的一个主要问题是数据稀疏问题。数据稀疏问题在基于统计的机器学习中是一个常见的问题，主要是由于训练样本不足而导致密度估计不准确。数据稀疏问题最直接的解决方法就是增加训练数据集的规模，但其边际效益会随着数据集规模的增加而递减。

平滑技术（Smoothing），即给一些没有出现的词组合赋予一定先验概率。除了加法平滑，还有很多平滑技术，比如 Good-Turing 平滑，Kneser-Ney 平滑等，其基本思想都是增加低频词的频率，而降低高频词的频率。

深度序列模型

假设一个神经网络$\mathrm{f}(\cdot,\theta)$，其输入为历史信息$h_t=x_{1:(t-1)}$，输出为词表 V 中的每个词$\mathrm{v}_k(1\leq k\leq|V|)$出现的概率，并满足

$$\sum_{k=1}^{|V|}f_k(x_{1:(t-1)},\theta)=1$$

条件概率$p_\theta\left(x_t|x_{1:(t-1)}\right)$可以从神经网络的输出中得到，

$$p_\theta\left(x_t|x_{1:(t-1)}\right)=f_{k_{x_t}}(x_{1:(t-1)},\theta)$$

嵌入层、特征层、输出层：利用神经网络模型来估计条件概率$p_\theta\left(x_t|x_{1:(t-1)}\right)$。

- **嵌入层**　令$\mathrm{h}_t=\mathrm{x}_{1:(t-1)}$表示输入的历史信息，一般为符号序列。假设词$x_t$对应词表中的索引为 k，则其 one-hot 向量表示为$\delta_t\in\{0,1\}^{|V|}$，即第 k 维为 1，其余为 0 的$|\mathrm{V}|$维向量。词x_t对应的向量表示为

$$e_t=M\delta_t=m_k$$

- **特征层**　特征层用于从输入向量序列$e_1,e_2,\cdots,e_{t-1}$中提取特征，输出为一个可以表示历史信息的向量h_t。

前馈网络模型和循环网络模型的不同之处在于循环神经网络利用隐藏状态来记录以前所有时刻的信息，而前馈神经网络只能接受前 n-1 个时刻的信息。

- **输出层**　输出层为一般使用 softmax 分类器，接受历史信息的向量表示$\mathrm{h}_t\in R^{d2}$输出为词

表中每个词的后验概率，输出大小为|V|。

参数学习

给定一个训练序列$\mathrm{x}_{1:T}$，深度序列模型的训练目标为找到一组参数θ，使得对数似然函数最大，再通过梯度上升法进行求解。

$$\log p_\theta(\mathrm{x}_{1:T}) = \sum_{t=1}^{T} log p_\theta(x_t|x_{1:(t-1)})$$

评价方法

困惑度

信息论的一个概念，可以用来衡量一个分布的不确定性。对于离散随机变量$\mathrm{x} \in \mathrm{X}$，其概率分布为 p(x)，困惑度为

$$2^{H(p)} = 2^{-\sum_{x \in X} p(x) \log_2 p(x)}$$

困惑度也可以用来衡量两个分布之间差异。对于一个未知的数据分布$p_r(x)$和一个模型分布$p_\theta(x)$，我们从$p_r(x)$中采样出一组测试样本$\mathrm{x}^{(1)}, \cdots, x^{(n)}$，模型分布$p_\theta(x)$的困惑度为

$$2^{H(\hat{p}_r, p_\theta)} = 2^{-\frac{1}{N}\sum_{n=1}^{N} \log_2 p_\theta(x^{(n)})}$$

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。因此模型分布$p_\theta(x)$的好坏可以用困惑度来评价。

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

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

模型 $p_\theta(\mathbf{x})$ 的困惑度定义为

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

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 $p_\theta(x_t^{(n)}|\mathbf{x}_{1:(t-1)}^{(n)})$ 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

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。

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

组合的精度（Precision），

由于精度只衡量生成序列中的 N 元组合是否在参考序列中出现，生成序列越短，其精度会越高，因此可以引入长度惩罚因子(Brevity Penalty)。如果生成序列的长度短于参考序列，就对其进行惩罚。

$$P_n(\mathbf{x}) = \frac{\sum_{w \in \mathcal{W}} \min\left(c_w(\mathbf{x}), \max_{k=1}^{K} c_w(\mathbf{s}^{(k)})\right)}{\sum_{w \in \mathcal{W}} c_w(\mathbf{x})}.$$

$$\mathrm{b(x)} = \begin{cases} 1, if\ l_x > l_s \\ \exp\left(1 - \frac{l_s}{l_x}\right), if\ l_x \le l_s \end{cases}$$

ROUGE

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 x，从真实数据分布中采样出的一组参考（Reference）序列$\mathbf{s}^{(1)}, \cdots, s^{(n)}$，令 W 为从参考序列中提取 N 元组合的集合，ROUGE-N 的定义为

$$\mathrm{ROUGE-N(x)} = \frac{\sum_{k=1}^{K} \sum_{w \in W} \min(c_w(x), c_w(s^{(k)}))}{\sum_{k=1}^{K} \sum_{w \in W} c_w(s^{(k)})}$$

序列生成模型中的学习问题

使用最大似然估计来学习自回归序列生成模型时，会存在以下三个主要问题：曝光偏差、训练目标不一致和计算效率。

曝光偏差问题

在自回归生成模型中，第 t 步的输入为模型生成的前缀序列$\hat{\mathbf{x}}_{1:t-1}$。而在训练时，我们使用的前缀序列是训练集中的真实数据$\mathbf{x}_{1:t-1}$，而不是模型预测的$\hat{\mathbf{x}}_{1:t-1}$。这种学习方式也称为教师强制（Teacher Forcing）。

曝光偏差（Exposure Bias）：模型生成的分布$p_\theta(x_{1:t-1})$和真实数据分布$p_r(x_{1:t-1})$并不严格一致，因此条件概率$p_\theta(x|x_{1:t-1})$在训练和测试会存在协变量偏移问题。一旦在预测前缀$\hat{\mathbf{x}}_{1:t-1}$的过程中存在错误，会导致错误传播，使得后续生成的序列也会偏离真实分布。

为了缓解曝光偏差的问题，我们可以在训练时混合使用真实数据和模型生成数据。在第 t 步时，模型随机使用真实数据x_{t-1}或前一步生成的词$\hat{x}_{t-1}$作为输入。一个较好的策略是在训练初期赋予 ϵ 较大的值，随着训练次数的增加逐步减少 ϵ 的取值。这种策略称为计划采样（Scheduled Sampling）。

训练目标不一致问题

序列生成模型一般是采用和任务相关的指标来进行评价，而训练时是使用最大似然估计，这导致训练目标和评价方法不一致。并且这些评价指标一般都是不可微的，无法直接使用基于梯度的方法来进行优化。

为了可以直接优化评价目标，我们可以将自回归的序列生成可以看作是一种马尔可夫决策过程，并使用强化学习的方法来进行训练。基于强化学习的序列生成模型不但可以解决训练和评价目标不一致问题，也可以有效地解决曝光偏差问题。

计算效率问题

在第 t 步时，前缀序列为$\mathrm{h}_t = \mathrm{x}_{1:(t-1)}$，词$\mathrm{x}_t$的条件概率为

$$p_\theta(x_t|h_t) = softmax\big(s(x_t, h_t; \theta)\big) = \frac{\exp(s(v, h_t; \theta))}{\sum_{v \in V} \exp(s(v, h_t; \theta))} = \frac{\exp(s(v, h_t; \theta))}{Z(h_t; \theta)}$$

层次化 Softmax

为了进一步降低 softmax 的计算复杂度，我们可以更深层的树结构来组织词汇表。假设用二叉树来组织词表中的所有词，二叉树的叶子节点代表词表中的词，非叶子节点表示不同层次上的类别。如果我们将二叉树上所有左链接标记为 0，右链接标记为 1。每一个词可以用根节点到它所在的叶子之间路径上的标记来进行编码。

假设词 v 在二叉树上从根节点到其所在叶子节点的路径长度为 m，其编码可以表示一个位向量（bit vector）：$[\mathrm{b}_1, \cdots, \mathrm{b}_m]^T$。词 v 的条件概率为

$$p(v|h) = p(\mathrm{b}_1, \cdots, \mathrm{b}_m|h) = \prod_{j=1}^{m} p(b_j|\mathrm{b}_1, \cdots, \mathrm{b}_{j-1}, h) = \prod_{j=1}^{m} p(b_j|b_{j-1}, h)$$

若使用平衡二叉树来进行分组，则条件概率估计可以转换为$\log_2 |V|$个两类分类问题。这时 softmax 函数可以用 logistic 函数代替，计算效率可以加速$\frac{|\mathrm{V}|}{\log_2 |V|}$倍。

- 利用人工整理的词汇层次结构，比如利用 WordNet 系统中的“IS-A”关系（即上下位关系）。因为 WordNet 的层次化结构不是二叉树，因此需要通过进一步聚类来转换为二叉树。
- 使用 Huffman 编码。Huffman 编码对出现概率高的词使用较短的编码，出现概率低的词则使用较长的编码。因此训练速度会更快。

算法 15.1: Huffman 树构建算法

输入: 词表: $\mathcal{V}$

初始化：为每个词 v 建立一个叶子节点，其概率为词的出现频率;
将所有的叶子节点放入集合 $\mathcal{S}$ 中;
while $|\mathcal{S}| > 1$ **do**
    从集合 $\mathcal{S}$ 选择两棵概率最低的节点 n_1 和 n_2;
    构建一个新节点 n'，并将 n_1 和 n_2 作为 n' 的左右子节点;
    新节点 n' 的概率 n_1 和 n_2 的概率之和，;
    将新二叉树加入集合 $\mathcal{S}$ 中，并把 n_1 和 n_2 从集合 $\mathcal{S}$ 中移除;
end
集合 $\mathcal{S}$ 中最后一个节点为 n;

输出: 以 n 为根节点的二叉树 T

重要性采样

通过采样来近似计算训练时的梯度。重要性采样的思想和算法都比较简单，但其效果依赖于建议分布$q(v|h_t)$的选取。如果$q(v|h_t)$选取不合适时，会造成梯度估计非常不稳定。在实践中，提议分布$q(v|h_t)$经常使用一元模型的分布函数。虽然直观上$q(v|h_t)$采用 N 元模型更加准确，但使用复杂的 N 元模型分布并不能改进性能，原因是 N 元模型的分布和神经网络模型估计的分布之间有很大的差异。

噪声对比估计

噪声对比估计是将密度估计问题转换为两类分类问题，从而降低计算复杂度。假设有三个分布，一个是需要建模真实数据分布$p_r(x)$；第二是模型分布$p_\theta(x)$，并期望调整模型参数θ来使得$p_\theta(x)$来拟合真实数据分布$p_r(x)$；第三个是噪声分布$q(x)$，用来对比学习。给定一个样本 x，

如果 x 是从$p_r(x)$中抽取的，称为真实样本，如果 x 是从q(x)中抽取的，则称为噪声样本。为了判断样本 x 是真实样本还是噪声样本，引入一个辨别函数 D。

一般噪声样本的数量要比真实样本大很多。为了提高近似效率，我们近似假设噪声样本的数量是真实样本的 K 倍，即 y 的先验分布满足以及贝叶斯公式知，

$$P(y=0)=KP(y=1)$$

$$p(x|y=0)=q(x), p(x|y=1)=p_\theta(x)$$

$$p(y=1|x)=\frac{p(x|y=1)p(y=1)}{p(x|y=0)p(y=0)+p(x|y=1)p(y=1)}=\frac{p_\theta(x)}{p_\theta(x)+Kq(x)}$$

依据噪声对比估计的损失函数，通过不断采样真实样本和噪声样本，并用梯度下降法可以学习参数θ使得$p_\theta(x)$逼近于真实分布$p_r(x)$。

$$\mathrm{L}(\theta)=-\frac{1}{N(K+1)}\left(\sum_{n=1}^{N} logp(y=1|x_n)+\sum_{n=1}^{KN} logp(y=0|x_n')\right)$$

噪声对比估计相当于用判别式的准则L(θ)来训练一个生成式模型$p_\theta(x)$，使得判别函数 D 很容易能分别出样本 x 来自哪个分布，其思想与生成式对抗网络类似。不同之处在于，在噪声对比估计中的判别函数 D 是通过贝叶斯公式计算得到，而生成对抗网络的判别函数 D 是一个需要学习的神经网络。

基于噪声对比估计的序列模型　虽然通过噪声对比估计，将一个|V|类的分类问题转为一个两类分类问题，但是依然需要计算$p_\theta(v|h)$，其中仍然涉及到配分函数的计算。为了避免计算配分函数，我们将负对数配分函数$-logZ(h,\theta)$作为一个可学习的参数z_h（即每一个 h 对应一个参数），这样条件概率$p_\theta(v|h)$重新定义为

$$p_\theta(v|h)=\exp(s(v,h;\theta))\exp(z_h)$$

噪声对比估计方法的一个特点是会促使未归一化分布$\exp(s(v,h;\theta))$可以自己学习到一个近似归一化的分布，并接近真实的数据分布$p_r(v|h)$。也就是说，学习出来的$\exp(z_h)\approx 1$。这样可以直接令$\exp(z_h)=1,\forall$h，并用未归一化的分布$\exp(s(v,h;\theta))$来代替$p_\theta(v|h)$。

$$p(y=1|v,h)=\frac{\exp(s(v,h;\theta))}{\exp(s(v,h;\theta))+Kq(v)}=\frac{1}{1+\frac{Kq(v)}{\exp(s(v,h;\theta))}}$$

$$=\frac{1}{1+\exp(-s(v,h;\theta)-\log(Kq(v)))}=\frac{1}{1+\exp(-\Delta s(v,h;\theta))}=\sigma(\Delta s(v,h;\theta))$$

基于采样的方法并不改变模型的结构，只是近似计算参数梯度。在训练时可以显著提高模型的训练速度，但是在测试阶段依然需要计算配分函数。而基于层次化 softmax 的方法改变了模型的结构，在训练和测试时都可以加快计算速度。

序列到序列模型

序列到序列模型的目标是估计条件概率

$$p_\theta(y_{1:t}|x_{1:s})=\prod_{t=1}^{T} p_\theta(y_t|y_{1:t-1},x_{1:s})$$

给定一组训练数据$\{(x_{S_n},y_{T_n})\}_{n=1}^{N}$，我们可以使用最大似然估计来训练模型参数。一旦训练完成，模型就可以根据一个输入序列 x 来生成最可能的目标序列，

$$\max_{\theta}\sum_{n=1}^{N} log p_{\theta}\left(y_{1:T_n}|x_{1:S_n}\right), \hat{y} = arg\max_{y} p_{\theta}(y|x)$$

基于循环神经网络的序列到序列模型

使用两个循环神经网络来分别进行编码和解码，也称为编码器-解码器（Encoder-Decoder）模型。

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

$$h_t^e = f_{enc}\left(h_{t-1}^e, e_{x_{t-1}}, \theta_{enc}\right), u = h_S^e$$

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

$$h_0^d = u, h_t^d = f_{dec}\left(h_{t-1}^d, e_{y_{t-1}}, \theta_{dec}\right), o_t = g(h_0^d, o_0)$$

基于循环神经网络的序列到序列模型的缺点是：（1）向量 c 的容量问题，输入序列的信息很难全部保存在一个固定维度的向量中；（2）当序列很长时，由于循环神经网络的长期依赖问题，容易丢失输入序列的信息。

基于注意力的序列到序列模型

在解码过程的第 t 步时，先用上一步的隐状态h_{t-1}^d作为查询向量，利用注意力机制从所有输入序列的隐状态$H^e = [h_1^e, \cdots, h_S^e]$中选择相关信息。

$$c_t = att\left(H^e, h_{t-1}^d\right) = \sum_{i=0}^{S} \alpha_i h_i^e = \sum_{i=0}^{S} softmax(s(h_i^e, h_{t-1}^d))h_i^e$$

$$h_t^d = f_{dec}\left(h_{t-1}^d, [e_{y_{t-1}}; c_t], \theta_{dec}\right)$$

基于自注意力的序列到序列模型

自注意力

对于一个向量序列$\mathrm{H} = [h_1, \cdots, h_T] \in R^{d_h \times T}$，首先用自注意力模型来对其进行编码。

$$self - att(Q, K, V) = softmax\left(\frac{K^T Q}{\sqrt{d_h}} V\right), Q = W_Q H, K = W_K X, V = W_V X$$

多头自注意力

为了提取更多的交互信息，我们可以使用多头注意力，在多个不同的投影空间中捕捉不同的交互信息。

$$\mathrm{MultiHead(H)} = W_O(head_1; \cdots; head_M), head_M = self - att(Q_m, K_m, V_m)$$
$$\forall m \in [1, M], Q_m = W_Q^m H, K = W_K^m X, V = W_V^m X$$

基于自注意力模型的序列编码

对于一个序列$x_{1:T}$，我们可以构建一个多层的多头自注意力来对其进行编码。由于自注意力模型忽略了输入信息的位置信息，因此初始的输入序列中加入位置编码信息来进行修正。对于一个输入序列$x_{1:T}$，

$$H^{(0)} = [e_{x_1} \oplus p_1, \cdots, e_{x_T} \oplus p_T]$$

第 1 层的隐状态$H^{(l)}$为

$$Z^{(l)} = norm(H^{(l-1)} + \text{MultiHead}(H^{(l-1)}))$$

$$H^{(l)} = norm(Z^{(l)} + \text{FFN}(Z^{(l)}))$$

$$\text{FFN}(z) = W_2 ReLu(W_1 z + b_1) + b_2$$

基于自注意力模型的序列编码可以看作是一个全连接的前馈神经网络，第 1 层的每个位置都接受第 l-1 层的所有位置的输出。不同的是，其连接权重是通过注意力机制动态计算得到。

基于自注意力模型的序列到序列模型

编码器 编码器只包含多层的自注意力模块，每一层都接受前一层的输出作为输入。编码器的输入为序列$x_{1:S}$，输出为一个向量序列$H^e = [h_1^e, \cdots, h_S^e]$。

解码器 解码器依是通过自回归的方式来生成目标序列。和编码器不同，解码器可以由以下三个模块构成：

- **自注意力模块：**第 t 步时，先使用自注意力模型对已生成的前缀序列$y_{1:(t-1)}$进行编码得到$H^d = [h_1^d, \cdots, h_{(t-1)}^d]$。在训练时，解码器的输入为整个目标序列，这时可以通过一个掩码（mask）来阻止每个位置选择其后面的输入信息。
- **解码器到编码器注意力模块：**使用$h_{(t-1)}^d$作为查询向量，通过注意力机制来从输入序列H^e中选取有用的信息。
- **逐位置的前馈神经网络：**使用一个前馈神经网络来综合得到所有信息。