

序列生成模型

<https://nndl.github.io/>

序列数据

- ▶ 在深度学习的应用中，有很多数据是以序列的形式存在，比如声音、语言、视频、DNA 序列或者其它的时序数据等。

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。
- ▶规则是什么？

语言模型

► 自然语言理解 → 一个句子的可能性 / 合理性

- ! 在报那猫告做只
- 那只猫在作报告!
- 那个人在作报告!



► 一切都是概率!

序列概率模型

► 给定一个序列样本，其概率为

$$p(x_{1:T}) = p(x_1, x_2, \dots, x_T)$$

► 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

$$p(x_{1:T}) = p(x_1, x_2, \dots, x_T)$$

- ▶ 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 $|V|^T$ 。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

$$\begin{aligned} p(x_{1:T}) &= \prod_t p(x_t | x_{1:t-1}) \\ &\approx \prod_t p(x_t | x_{t-1}, \dots, x_{t-n+1}) = \prod_t g(h_t) \end{aligned}$$

- 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 $p(x_t | x_{1:t-1})$ 。

序列概率模型

► 序列概率

- ▶ $p(x_1, x_2, \dots, x_T)$
- ▶ $= \prod_t p(x_t | x_{t-1}, \dots, x_1)$
- ▶ $\approx \prod_t p(x_t | x_{t-1}, \dots, x_{t-n+1}) = \prod_t g(h_t)$

- ▶ 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，

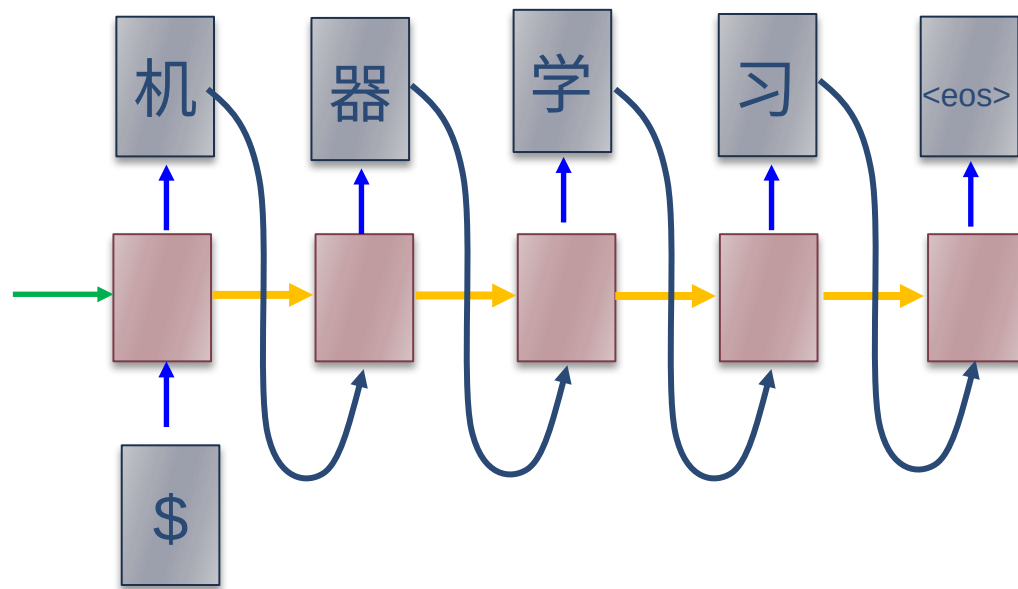
$$\hat{x}_t \sim p_\theta(x|\hat{\mathbf{x}}_{1:(t-1)}), \quad (15.6)$$

其中 $\hat{\mathbf{x}}_{1:(t-1)} = \hat{x}_1, \dots, \hat{x}_{t-1}$ 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。

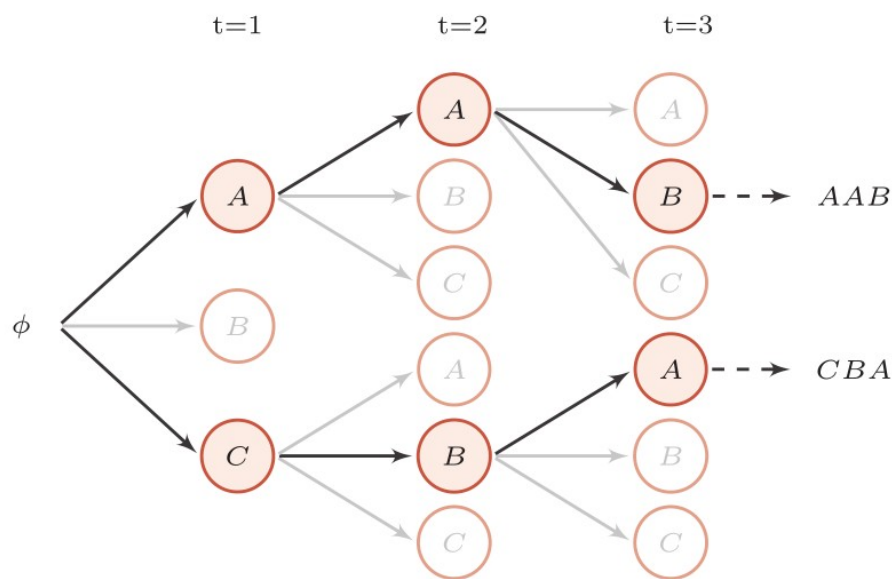
生成最可能序列

- ▶ 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。

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

束搜索

- ▶ 一种常用的减少搜索错误的启发式方法是束搜索（Beam Search）。
- ▶ 在每一步的生成中，生成 K 个最可能的前缀序列，其中 K 为束的大小（Beam Size），是一个超参数。



N 元统计模型

由于数据稀疏问题，当 t 比较大时，依然很难估计条件概率 $p(x_t|\mathbf{x}_{1:(t-1)})$ 。一个简化的方法是 N 元模型（N-Gram Model），假设每个词 x_t 只依赖于其前面的 $n - 1$ 个词（ n 阶马尔可夫性质），即

$$p(x_t|\mathbf{x}_{1:(t-1)}) = p(x_t|\mathbf{x}_{(t-n+1):(t-1)}). \quad (15.9)$$

当 $n = 1$ 时，称为一元（unigram）模型；当 $n = 2$ 时，称为二元（bigram）模型，以此类推。

$$p(x_t|\mathbf{x}_{(t-n+1):(t-1)}) = \frac{\mathbf{m}(\mathbf{x}_{(t-n+1):t})}{\mathbf{m}(\mathbf{x}_{(t-n+1):(t-1)})},$$

平滑技术

- ▶ N 元模型的一个主要问题是数据稀疏问题。
 - ▶ 数据稀疏问题的一种解决方法是平滑技术（Smoothing），即给一些没有出现的词组合赋予一定先验概率。平滑技术是 N 元模型中的一项必不可少的技术，比如加法平滑的计算公式为：

$$p(x_t | \mathbf{x}_{(t-n+1):(t-1)}) = \frac{m(\mathbf{x}_{(t-n+1):t}) + \delta}{m(\mathbf{x}_{(t-n+1):(t-1)}) + \delta|\mathcal{V}|},$$

- ▶ $\delta = 1$ 时，称为加 1 平滑。

深度序列模型

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

$$\sum_{k=1}^{|\mathcal{V}|} f_k(\mathbf{x}_{1:(t-1)}, \theta) = 1, \quad (15.21)$$

其中 θ 表示网络参数。条件概率 $p_\theta(x_t | \mathbf{x}_{1:(t-1)})$ 可以从神经网络的输出中得到，

$$p_\theta(x_t | \mathbf{x}_{1:(t-1)}) = f_{k_{x_t}}(\mathbf{x}_{1:(t-1)}; \theta), \quad (15.22)$$

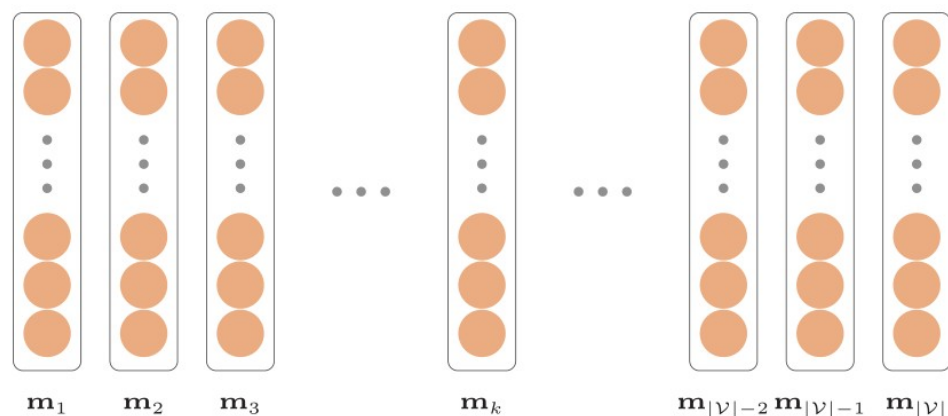
其中 k_{x_t} 为 x_t 在词表 $\mathcal{V}$ 中索引。

深度序列模型一般可以分为三个部分：嵌入层、特征层、输出层。

嵌入层

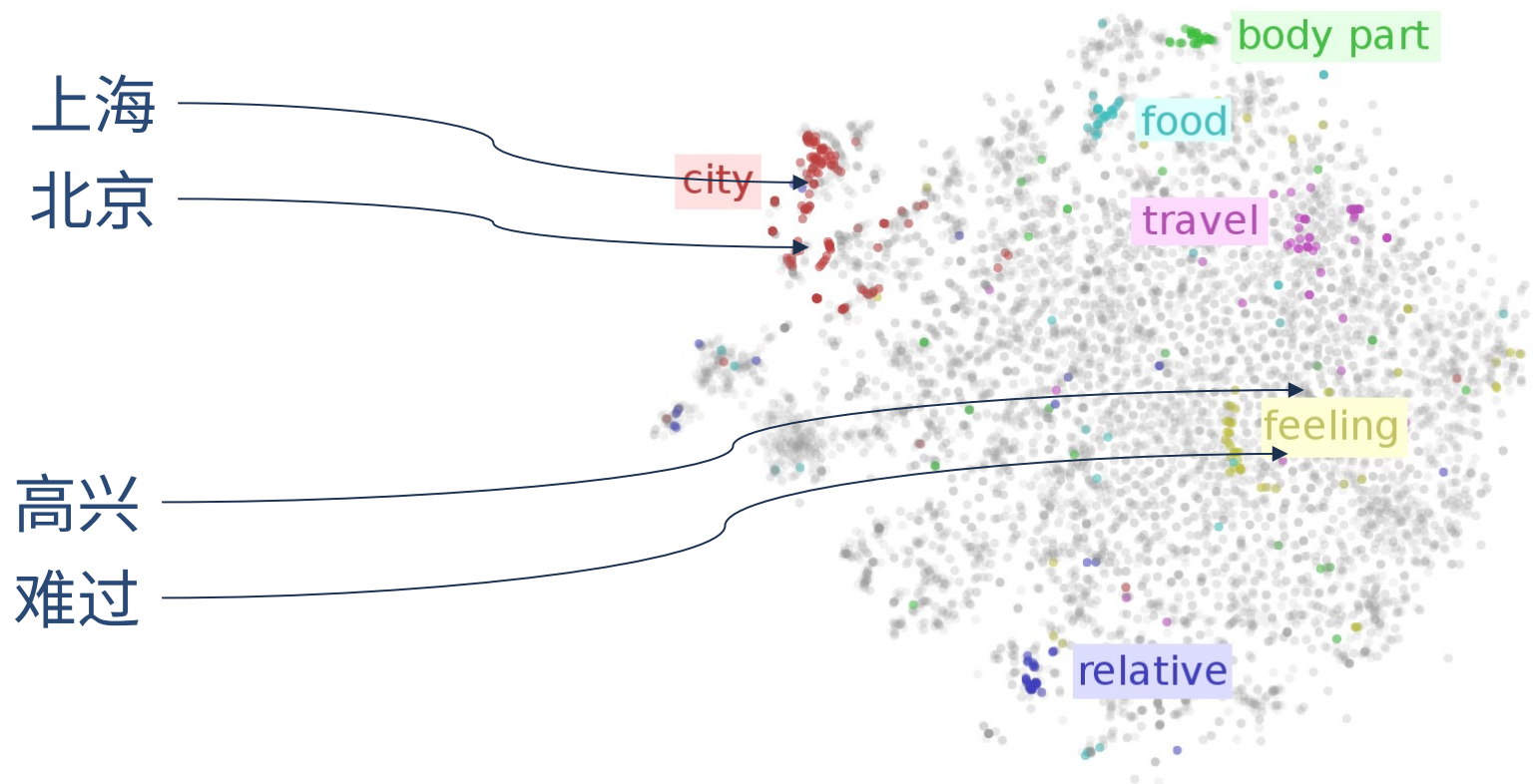
假设词 x_t 对应词表中的索引为 k ，则其 one-hot 向量表示为 $\delta_t \in \{0, 1\}^{|\mathcal{V}|}$ ，即第 k 维为 1，其余为 0 的 $|\mathcal{V}|$ 维向量。词 x_t 对应的向量表示为

$$\mathbf{e}_t = M\delta_t = \mathbf{m}_k. \quad (15.23)$$



通过上面的映射可以得到序列 $x_{1:(t-1)}$ 对应的向量序列 $\mathbf{e}_1, \dots, \mathbf{e}_{t-1}$ 。

词嵌入 (Word Embeddings)



<https://indico.io/blog/visualizing-with-t-sne/>

特征层

- ▶ 特征层可以通过不同类型的神经网络来实现，比如前馈神经网络和循环神经网络。
- ▶ 常见的网络类型有以下三种：
 - ▶ 简单平均
 - ▶ 前馈神经网络
 - ▶ 循环神经网络

特征层：简单平均

► 历史信息的平均

(1) 简单平均

$$\mathbf{h}_t = \sum_{i=1}^{t-1} \alpha_i \mathbf{e}_i, \quad (15.24)$$

其中 α_i 为每个词的权重。

权重 α_i 可以和位置 i 及其表示 $\mathbf{e}_i$ 相关，也可以无关。简单起见，可以设置 $\alpha_i = \frac{1}{t-1}$ 。权重 α_i 也可以通过注意力机制来动态计算。

特征层：前馈神经网络

(2) 前馈神经网络

前馈神经网络要求输入的大小是固定的。因此和N元模型类似，假设历史信息只包含前面 $n - 1$ 个词。

首先将这 $n - 1$ 个的词向量的 $\mathbf{e}_{t-n+1}, \dots, \mathbf{e}_{t-1}$ 拼接成一个 $d_1 \times (n - 1)$ 维的向量 $\mathbf{h}'$ 。

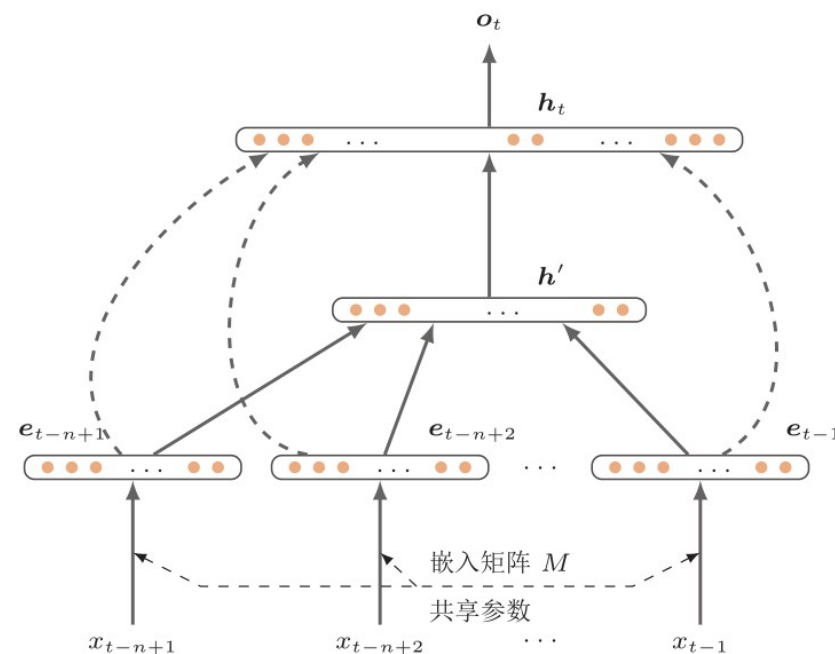
$$\mathbf{h}' = \mathbf{e}_{t-n+1} \oplus \dots \oplus \mathbf{e}_{t-1}, \quad (15.25)$$

其中 $\oplus$ 表示向量拼接操作。

然后将 $\mathbf{h}'$ 输入到由前馈神经网络构成的隐藏层，最后一层隐藏层的输出 $\mathbf{h}_t$ 。

$$\mathbf{h}_t = g(\mathbf{h}', \theta_g), \quad (15.26)$$

其中 $g(\cdot, \theta_g)$ 可以为全连接的前馈神经网络或卷积神经网络， θ_g 为网络参数。



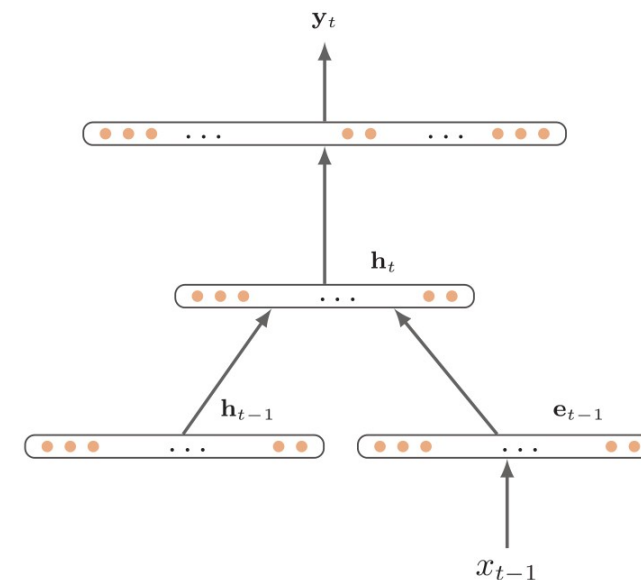
特征层：循环网络

(3) 循环神经网络

和前馈神经网络不同，循环神经网络可以接受变长的输入序列，依次接受输入 $\mathbf{e}_1, \dots, \mathbf{e}_{t-1}$ ，得到时刻 t 的隐藏状态 $\mathbf{h}_t$

$$\mathbf{h}_t = g(\mathbf{h}_{t-1}, \mathbf{e}_t, \theta_g), \quad (15.28)$$

其中 $g(\cdot)$ 为一个非线性函数， θ_g 为循环网络的参数， $\mathbf{h}_0 = 0$ 。



- ▶ 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，输出为词表中每个词的后验概率。

$$\begin{aligned}\mathbf{o}_t &= \text{softmax}(\hat{\mathbf{o}}_t) \\ &= \text{softmax}(W\mathbf{h}_t + \mathbf{b}),\end{aligned}$$

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

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

模型 $p_\theta(\mathbf{x})$ 的困惑度定义为

$$\text{PPL}(\theta) = 2^{-\frac{1}{T} \sum_{n=1}^N \log p_\theta(\mathbf{x}_{1:T_n}^{(n)})} \quad (15.36)$$

$$= 2^{-\frac{1}{T} \sum_{n=1}^N \sum_{t=1}^{T_n} \log p_\theta(x_t^{(n)} | \mathbf{x}_{1:(t-1)}^{(n)})} \quad (15.37)$$

$$= \left(\prod_{n=1}^N \prod_{t=1}^{T_n} p_\theta(x_t^{(n)} | \mathbf{x}_{1:(t-1)}^{(n)}) \right)^{-1/T}, \quad (15.38)$$

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

- ▶ 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 $\mathbf{s}^{(1)}, \dots, \mathbf{s}^{(K)}$ ，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），

$$P_n(\mathbf{x}) = \frac{\sum_{w \in \mathcal{W}} \min(c_w(\mathbf{x}), \max_{k=1}^K c_w(\mathbf{s}^{(k)}))}{\sum_{w \in \mathcal{W}} c_w(\mathbf{x})}, \quad (15.39)$$

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。

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

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

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 $\mathbf{s}^{(1)}, \dots, \mathbf{s}^{(K)}$ ，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

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

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。

序列到序列模型

序列到序列模型

序列到序列（Sequence-to-Sequence, Seq2Seq）是一种条件的序列生成问题，给定一个序列 $\mathbf{x}_{1:S}$ ，生成另一个序列 $\mathbf{y}_{1:T}$ 。输入序列的长度 S 和输出序列的长度 T 可以不同。比如在机器翻译中，输入源语言，输出为目标语言。图15.5给出了基于循环神经网络的序列到序列机器翻译示例。

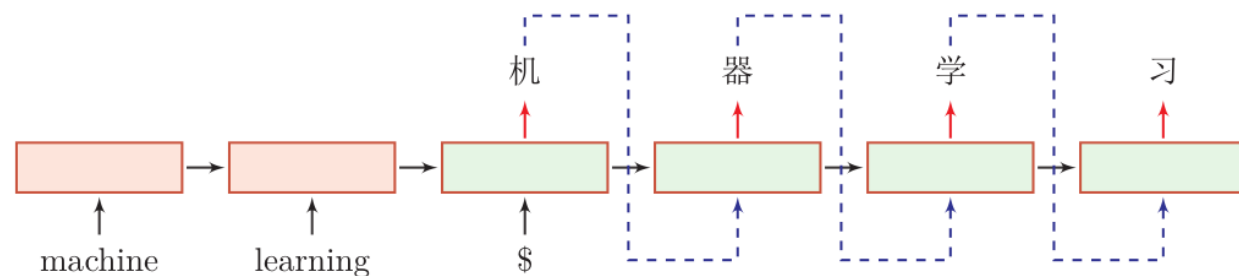


图 15.5 基于循环神经网络的序列到序列机器翻译

序列到序列模型

序列到序列模型的目标是估计条件概率

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

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。

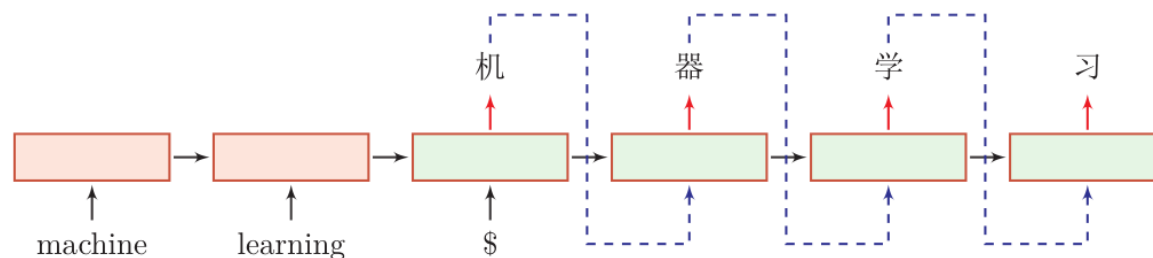
$$\max_{\theta} \sum_{n=1}^N \log p_{\theta}(\mathbf{y}_{1:T_n}|\mathbf{x}_{1:S_n}). \quad (15.97)$$

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，

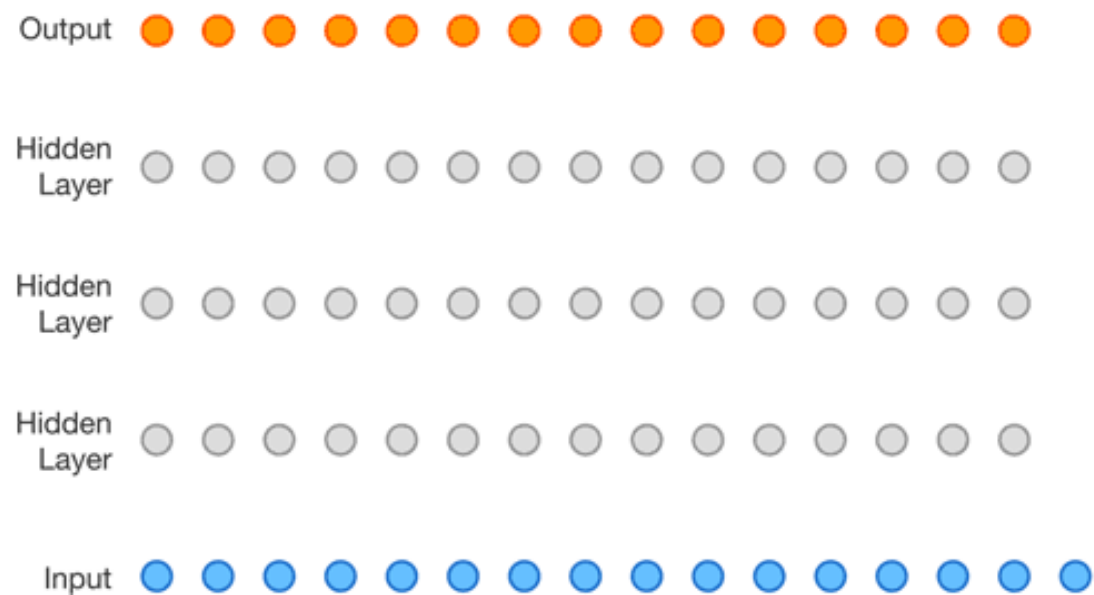
$$\hat{\mathbf{y}} = \arg \max_{\mathbf{y}} p_{\theta}(\mathbf{y}|\mathbf{x}), \quad (15.98)$$

具体的生成过程可以通过贪婪方法或束搜索来完成。

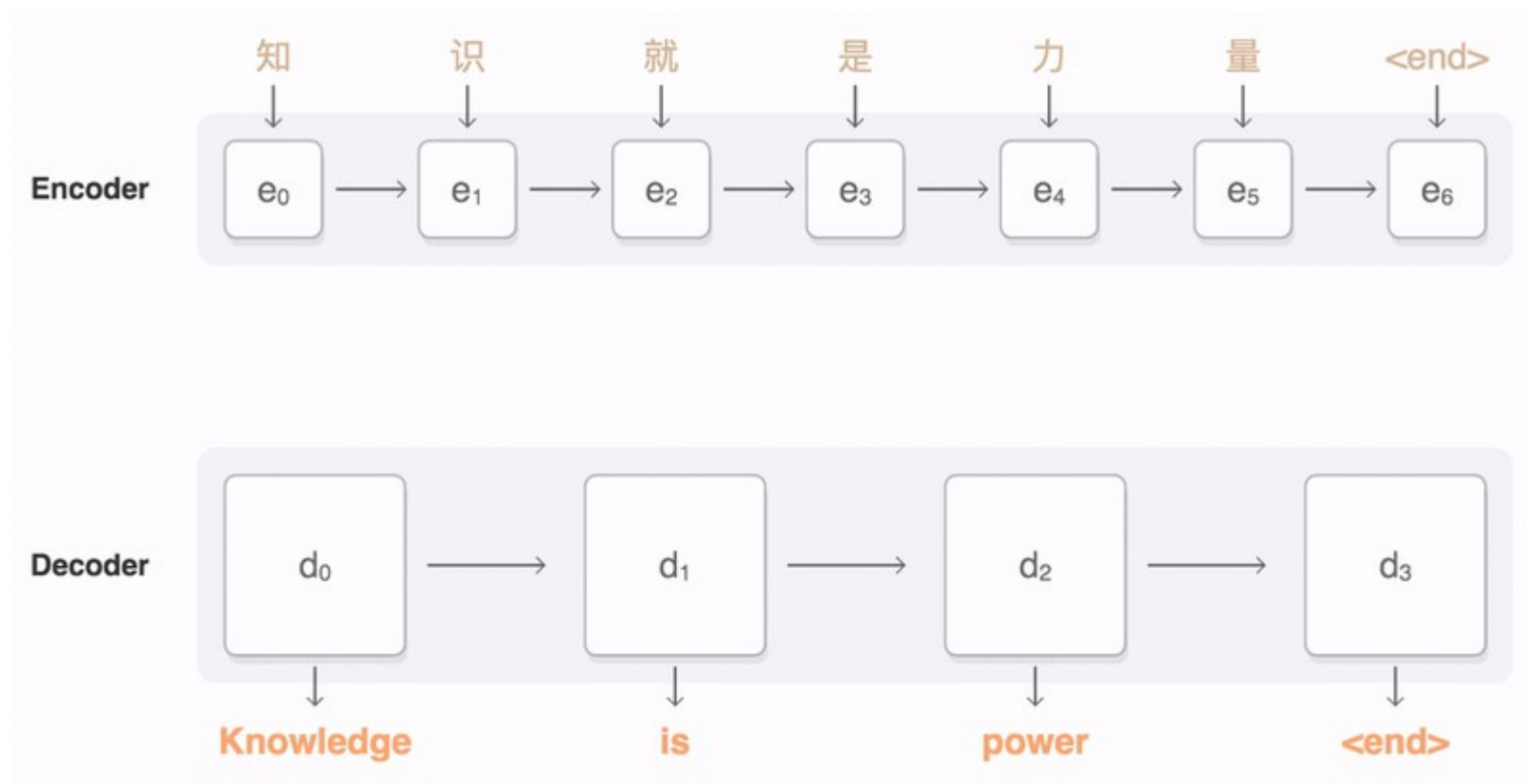
基于循环神经网络的序列到序列模型



基于前馈神经网络的序列到序列模型



基于注意力的序列到序列模型



基于卷积神经网络的序列到序列模型



基于自注意力的序列到序列模型

回顾：注意力模型

► 软性注意力机制 (soft attention mechanism)

► 注意力机制可以分为两步

► 计算注意力分布 α ,

$$\alpha_i = p(z = i | X, \mathbf{q})$$

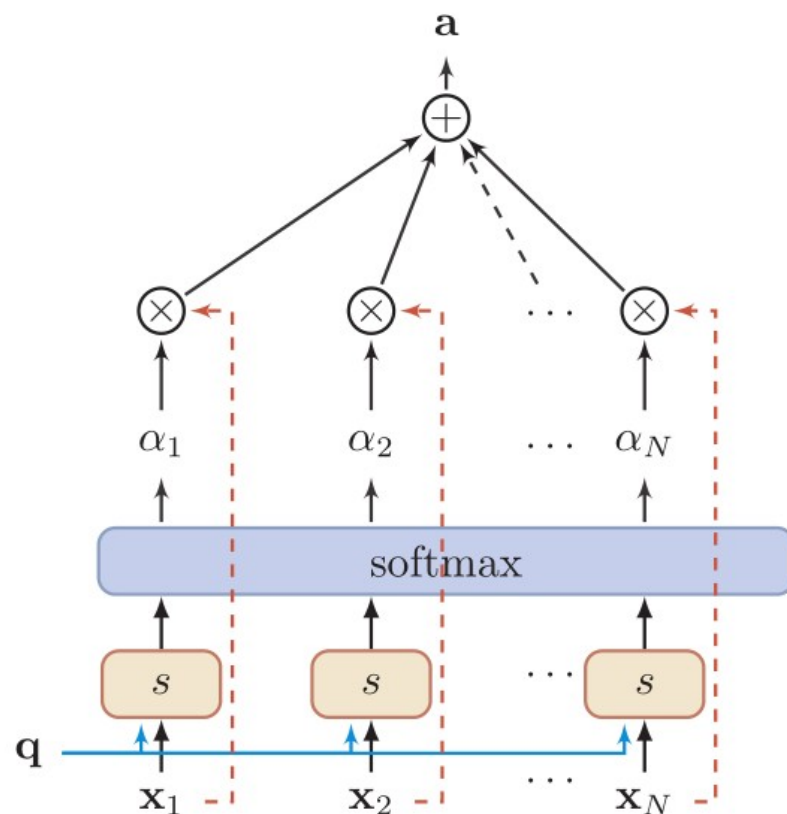
$$= \text{softmax}(s(\mathbf{x}_i, \mathbf{q}))$$

$$= \frac{\exp(s(\mathbf{x}_i, \mathbf{q}))}{\sum_{j=1}^N \exp(s(\mathbf{x}_j, \mathbf{q}))}$$

$s(\mathbf{x}_i, \mathbf{q})$ 打分函数

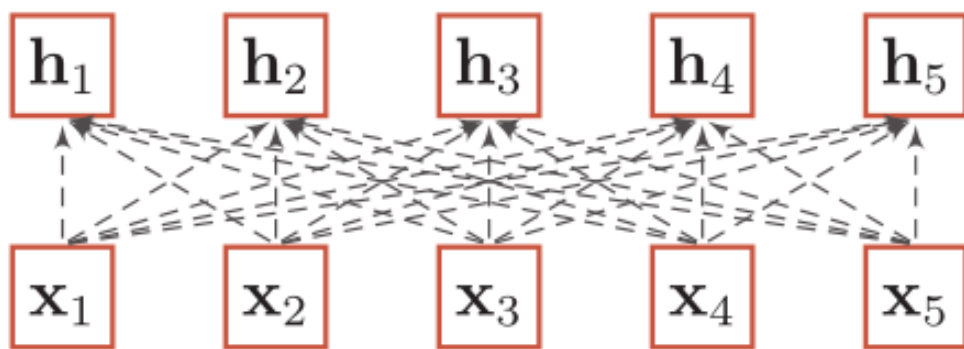
► 根据 α 来计算输入信息的加权平均。

$$\begin{aligned} \text{att}(X, \mathbf{q}) &= \sum_{i=1}^N \alpha_i \mathbf{x}_i, \\ &= \mathbb{E}_{z \sim p(z | X, \mathbf{q})}[\mathbf{x}] \end{aligned}$$

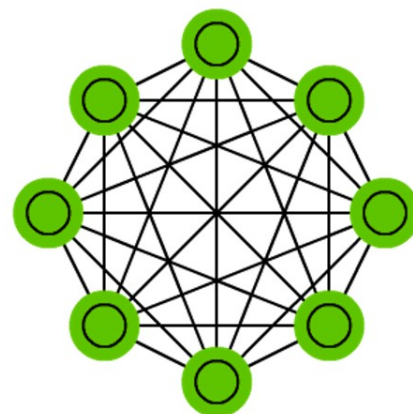


回顾：自注意力

► 基于自注意力的全连接神经网络

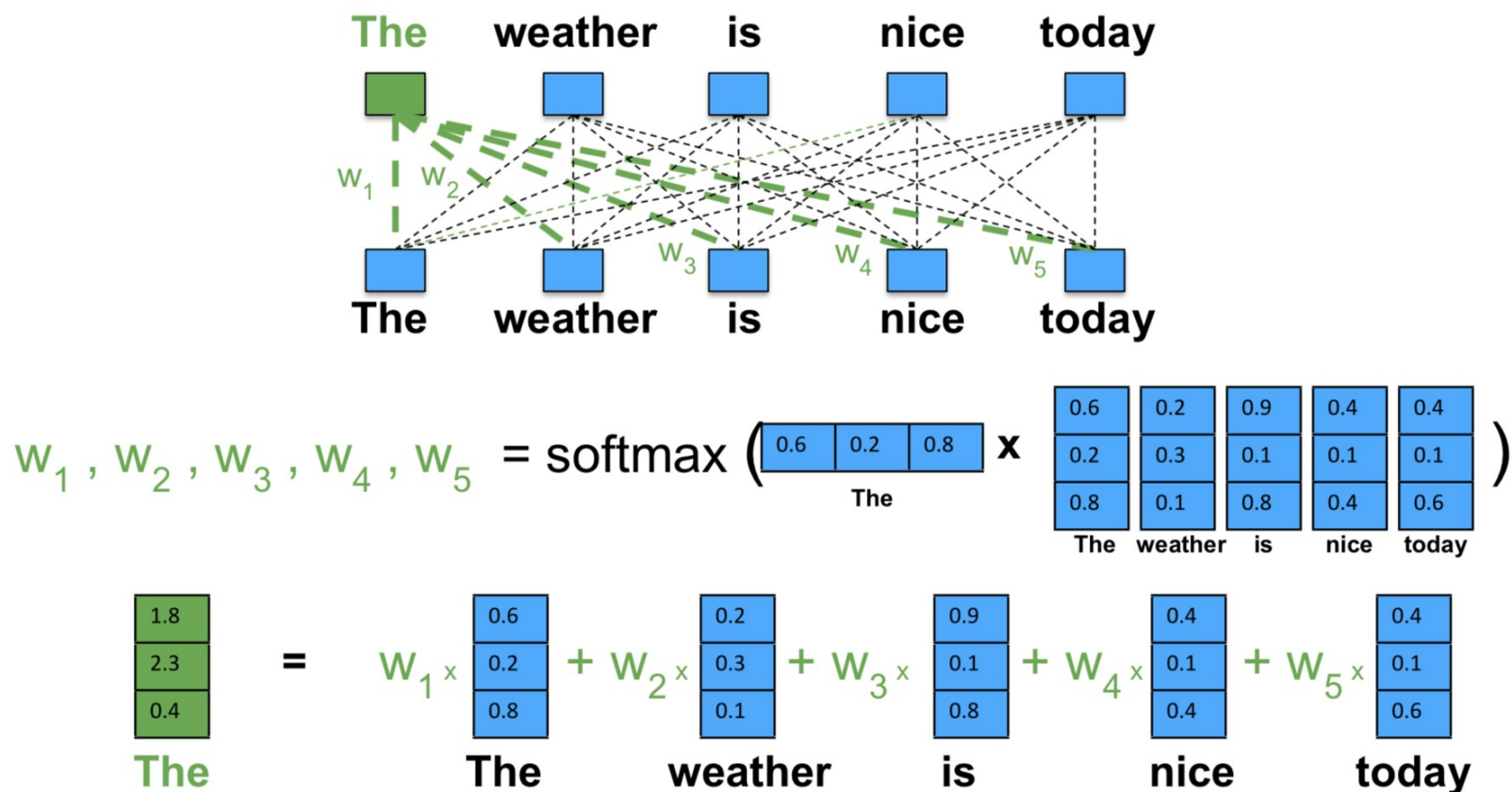


连接权重 α_{ij} 由注意力机制动态生成



也可以看作是一种全连接的图神经网络

回顾：自注意力示例



图片来源: http://fuyw.top/NLP_02_QANet/

QKV 模式 (Query-Key-Value)

Thinks
Machines

$$\begin{matrix} \text{X} \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline & & & \\ \hline \end{array} \end{matrix} \times \begin{matrix} W^Q \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline & & & \\ \hline & & & \\ \hline \end{array} \end{matrix} = \begin{matrix} Q \\ \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline \end{array} \end{matrix}$$

Thinks
Machines

$$\begin{matrix} \text{X} \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline & & & \\ \hline \end{array} \end{matrix} \times \begin{matrix} W^K \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline & & & \\ \hline & & & \\ \hline \end{array} \end{matrix} = \begin{matrix} K \\ \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline \end{array} \end{matrix}$$

Thinks
Machines

$$\begin{matrix} \text{X} \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline & & & \\ \hline \end{array} \end{matrix} \times \begin{matrix} W^V \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline & & & \\ \hline & & & \\ \hline \end{array} \end{matrix} = \begin{matrix} V \\ \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline \end{array} \end{matrix}$$

$$\text{softmax} \left(\frac{\begin{matrix} Q \\ \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline \end{array} \end{matrix} \times \begin{matrix} K^T \\ \begin{array}{|c|c|} \hline & \\ \hline & \\ \hline & \\ \hline \end{array} \end{matrix}}{\sqrt{d_k}} \right) \begin{matrix} V \\ \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline \end{array} \end{matrix}$$
$$= \begin{matrix} Z \\ \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline \end{array} \end{matrix}$$

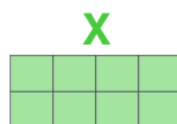
图片来源: <http://jalammar.github.io/illustrated-transformer/>

多头（multi-head）自注意力模型

1) This is our input sentence*

Thinking
Machines

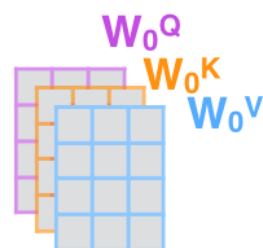
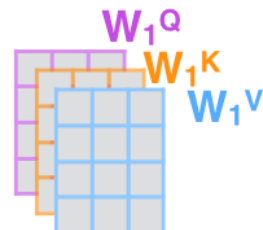
2) We embed each word*



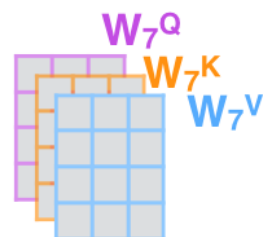
* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one



3) Split into 8 heads. We multiply X or R with weight matrices



...



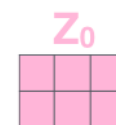
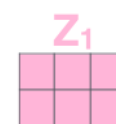
4) Calculate attention using the resulting $Q/K/V$ matrices



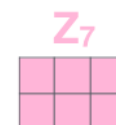
...



5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer



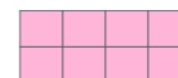
...



W^O



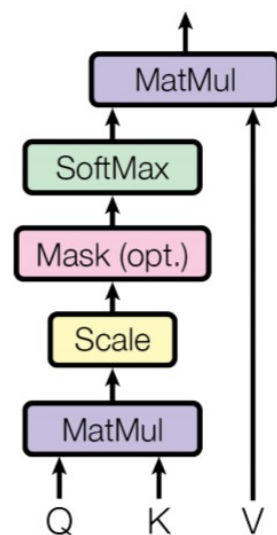
Z



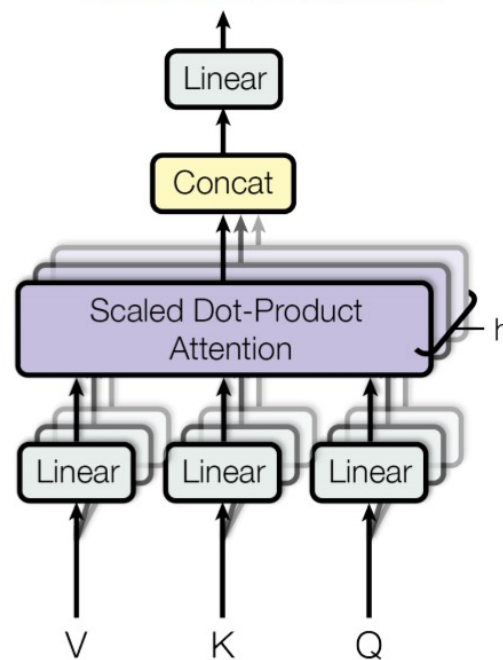
图片来源: <http://jalammr.github.io/illustrated-transformer/>

Transformer

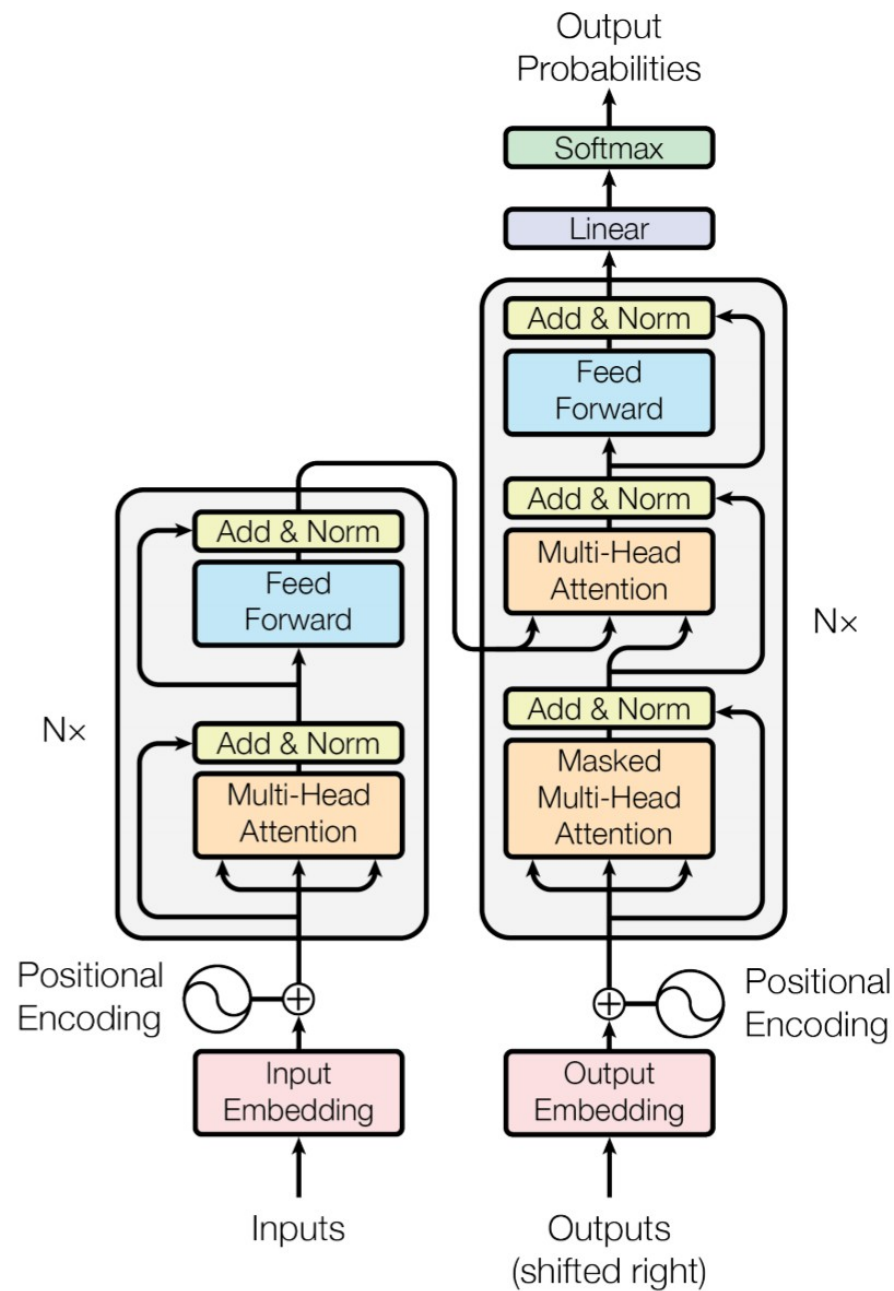
Scaled Dot-Product Attention



Multi-Head Attention

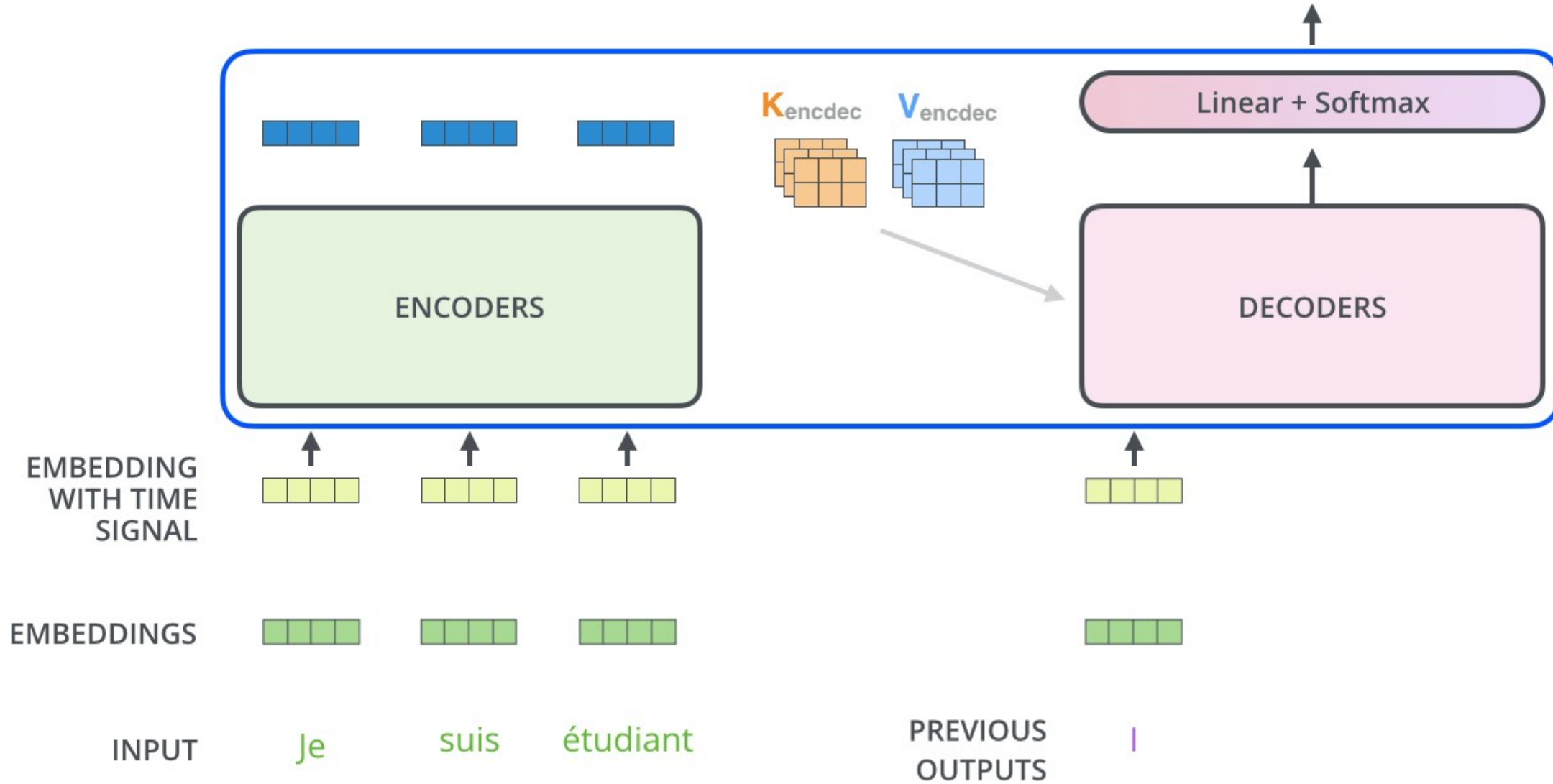


Transformer



Decoding time step: 1 2 3 4 5 6

OUTPUT |

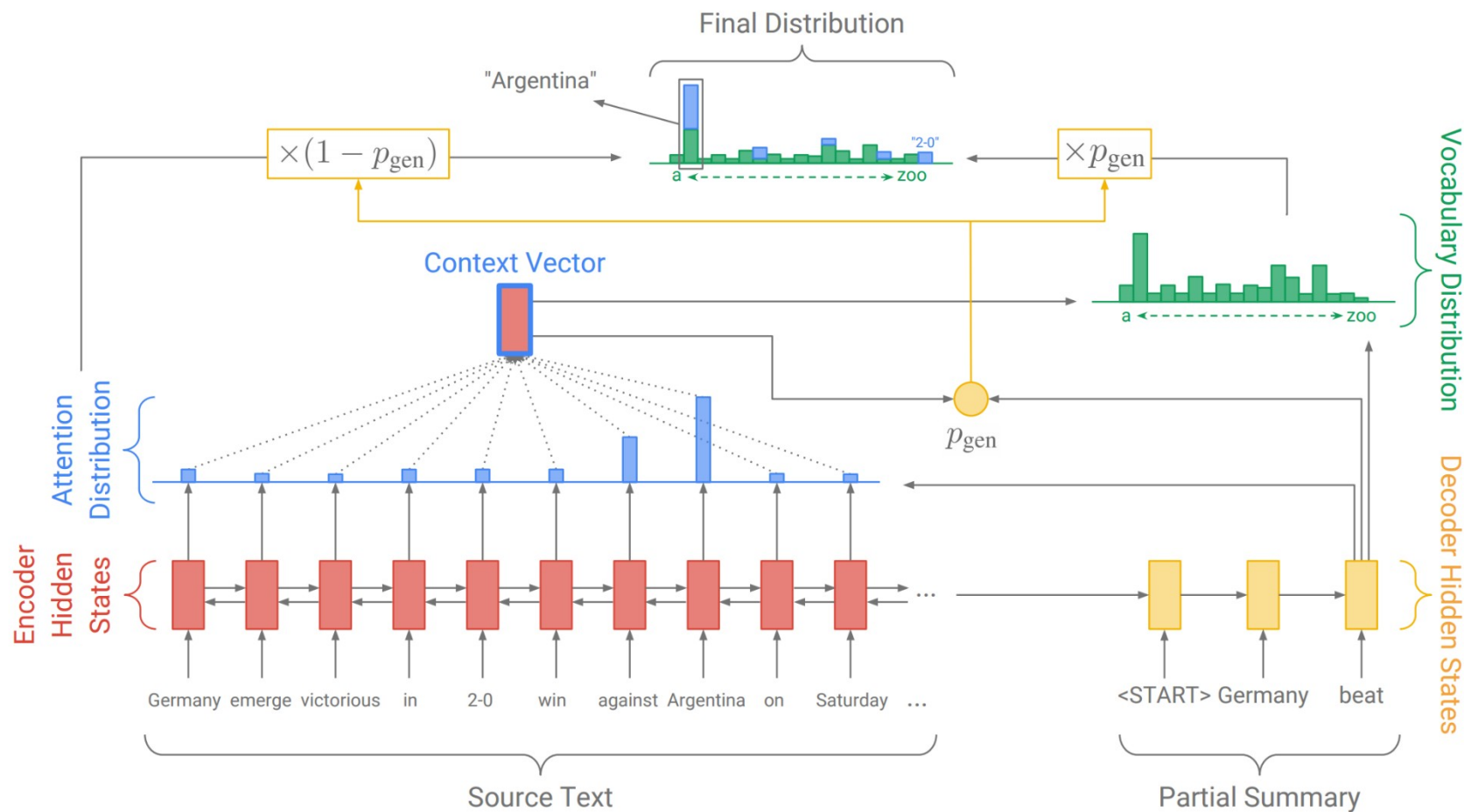


基于 Transformer 的序列到序列模型

其它应用

文本摘要

<http://www.abigailsee.com/2017/04/16/taming-rnns-for-better-summarization.html>



Reference summary

utility back francis *saili* will join up with munster later this year .
the new zealand international has signed a two-year contract .
saili made his debut for the all blacks against argentina in 2013 .

Sequence-to-sequence + attention summary

dutch international francis [UNK] has signed a two-year deal to join irish [UNK] super rugby side the blues .
[UNK] 's signature is something of a coup for munster and his head coach anthony foley believes he will be a great addition to their respective prospects .
[UNK] has been capped twice by new zealand .

Pointer-generator summary

new zealand international francis *saili* will move to the province later this year .
utility back *saili* made his all blacks debut against argentina in 2013 .
utility back *saili* will move to the province later this year .

Pointer-generator model + coverage summary

francis *saili* has signed a two-year deal to join munster later this year .
the 24-year-old was part of the new zealand under-20 side that won the junior world championship in italy in 2011 .
saili 's signature is something of a coup for munster and head coach anthony foley .

对话

Li J, Monroe W, Ritter A, et al. Deep reinforcement learning for dialogue generation[J]. arXiv preprint arXiv:1606.01541, 2016.

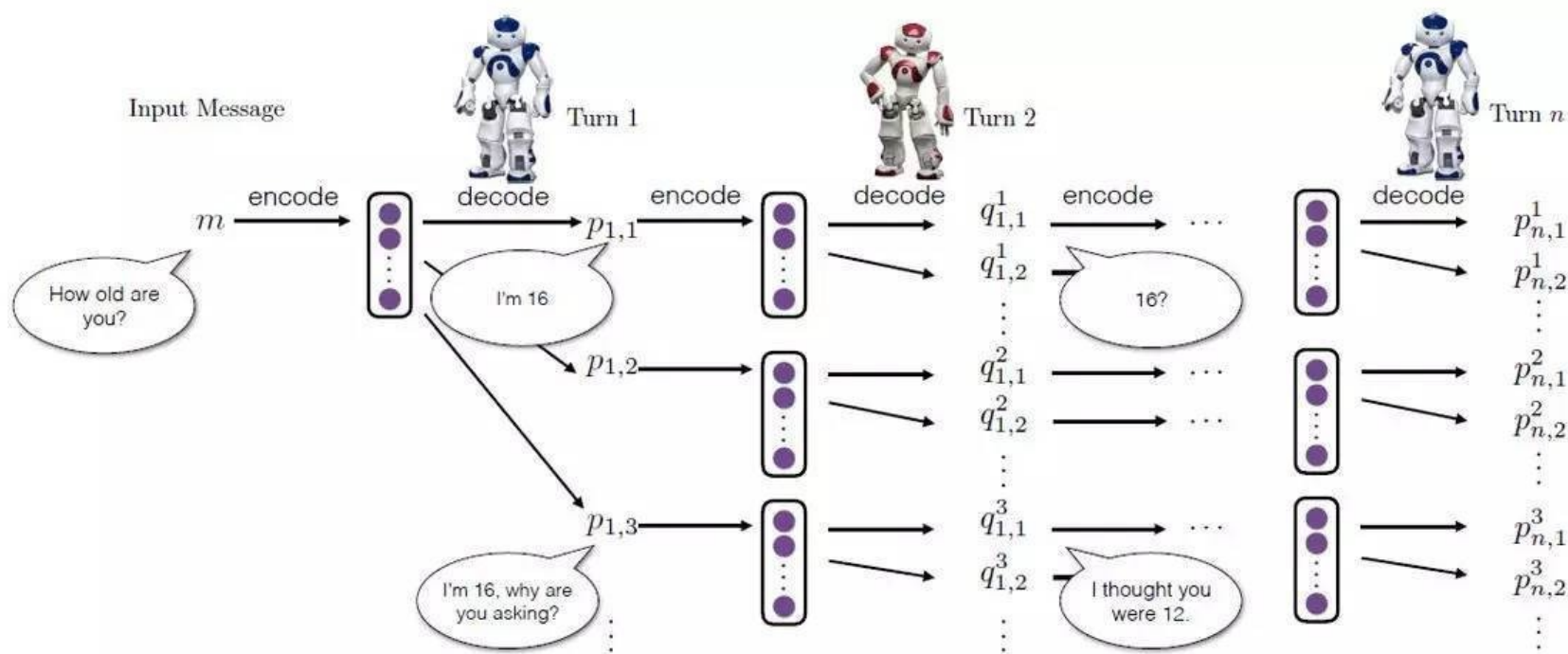
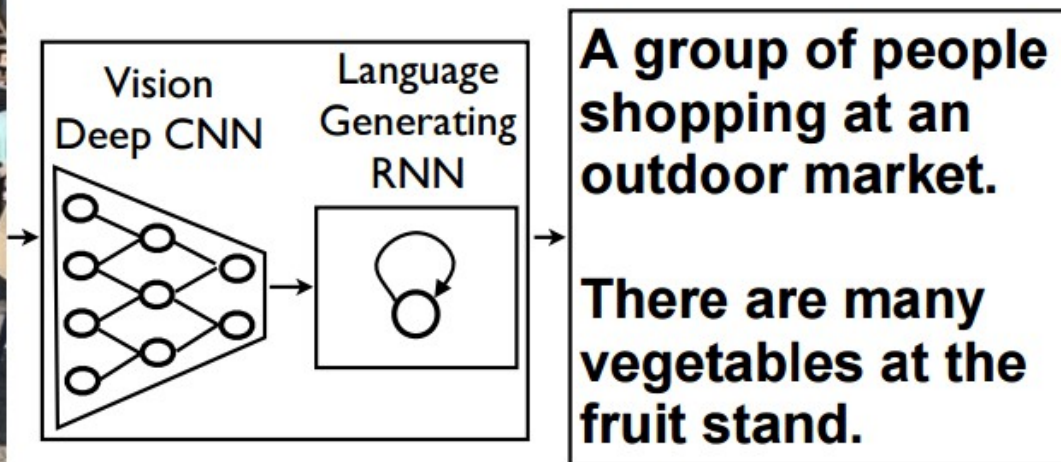


Figure 1: Dialogue simulation between the two agents.

看图说话



看图说话



Figure 5. A selection of evaluation results, grouped by human rating.

作诗

<p>白鹭窥鱼立， Egrets stood, peeping fishes. 青山照水开。 Water was still, reflecting mountains. 夜来风不动， The wind went down by nightfall, 明月见楼台。 as the moon came up by the tower.</p>	<p>满怀风月一枝春， Budding branches are full of romance. 未见梅花亦可人。 Plum blossoms are invisible but adorable. 不为东风无此客， With the east wind comes Spring. 世间何处是前身。 Where on earth do I come from?</p>
--	--

谢 谢

<https://nndl.github.io/>