循环神经网络

序列模型

序列模型指的是对一系列有时序或其他自然排序方式的变量进行建模，这类数据在自然语言处理，音频处理，时间序列分析等领域非常常见。序列模型的目标是基于输入序列的特征来预测输出序列，常见模型包括**马尔可夫模型(HMM)**，**循环神经网络(RNN)**以及其变体**长短期记忆网络(LSTM)**和**门控循环单元(GRU)**。

序列模型的关键在于捕捉和利用数据的**时序信息**以及**上下文依赖性**，这是处理**连续流数据**的重要能力。

假设在时间t观察到X_T，那么得到了T个**不独立的**随机变量服从分布$p(X)$， 表示为$(X_1, X_2, \ldots, X_T) \sim p(X)$，即$p(X_1, X_2, \ldots, X_T) = p(X)$。通过条件概率公式，我们可以将这个等式的表达为：

$$p(X) = p(x_1) \cdot p(x_2|x_1) \cdot p(x_3|x_1, x_2) \cdot \cdots \cdot p(x_T|x_1, \ldots x_{T-1}) \tag{1}$$

条件概率建模

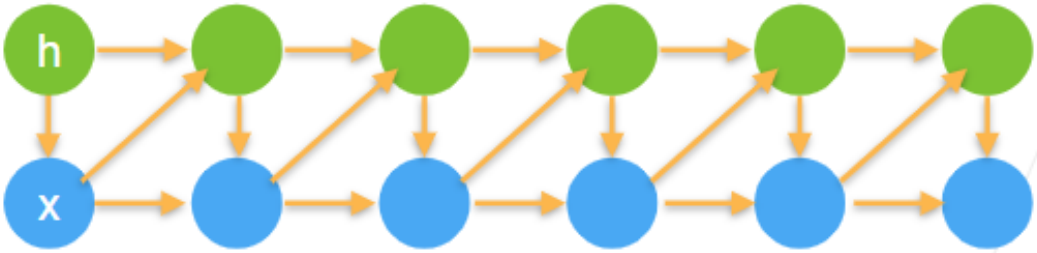
这个建模方法是对见过的数据进行建模，也称**自回归模型**。

- ***马尔可夫假设***

  假设当前数据**只跟过去的τ个数据点相关**

- ***潜变量模型***

  使用潜变量h_t来表示过去的信息，表示为$h_t = f(x_1, \ldots x_{t-1})$。然后用潜变量来预测未来信息，表示为$\hat{x}_t = p(x_t|h_t)$。

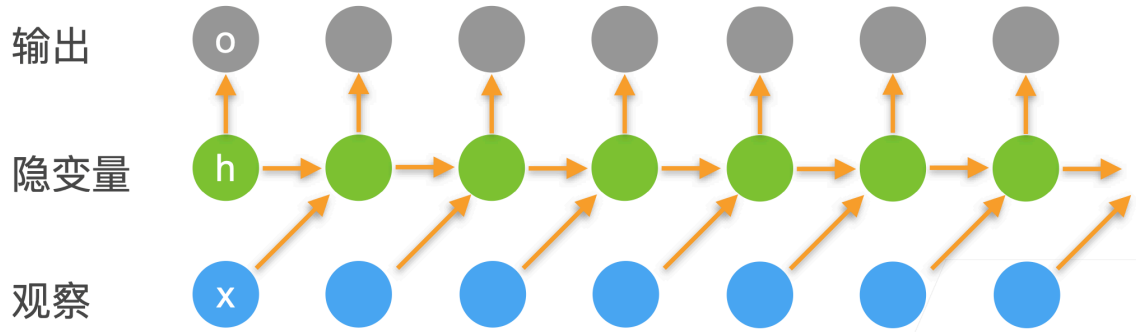


循环神经网络

循环神经网络是一种潜变量模型，它使用隐藏层（隐藏状态）来记录过去所有时间的信息。这样做可以在保留先前序列信息的避免计算所有信息带来的巨大计算量。

流程

1. 对于时刻t，我们使用t-1时刻的输入x_{t-1}以及潜变量h_{t-1}来计算新潜变量h_t。
2. 使用计算得到的新潜变量h_t以及t时刻新的输入x_t计算得到t时刻的输出o_t。



- 更新隐藏状态： $\mathbf{h}_t = \phi(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{hx}\mathbf{x}_{t-1} + \mathbf{b}_h)$
- 输出： $\mathbf{o}_t = \phi(\mathbf{W}_{ho}\mathbf{h}_t + \mathbf{b}_o)$

去掉这一项就退化成MLP

其中，**隐变量**的状态为：

$$h_t = \phi(h_{t-1}W_{hh} + x_{t-1}W_{xh} + b_h) \tag{2}$$

模型的**输出**为：

$$o_t = \phi(h_tW_{ho} + b_o) \tag{3}$$

如果没有考虑上一时刻的隐变量影响，RNN将成为单隐藏层的**MLP**模型。换句话说，循环神经网络的核心在于模型使用前一时刻的隐变量将之前的序列信息考虑进来。

评判标准：困惑度

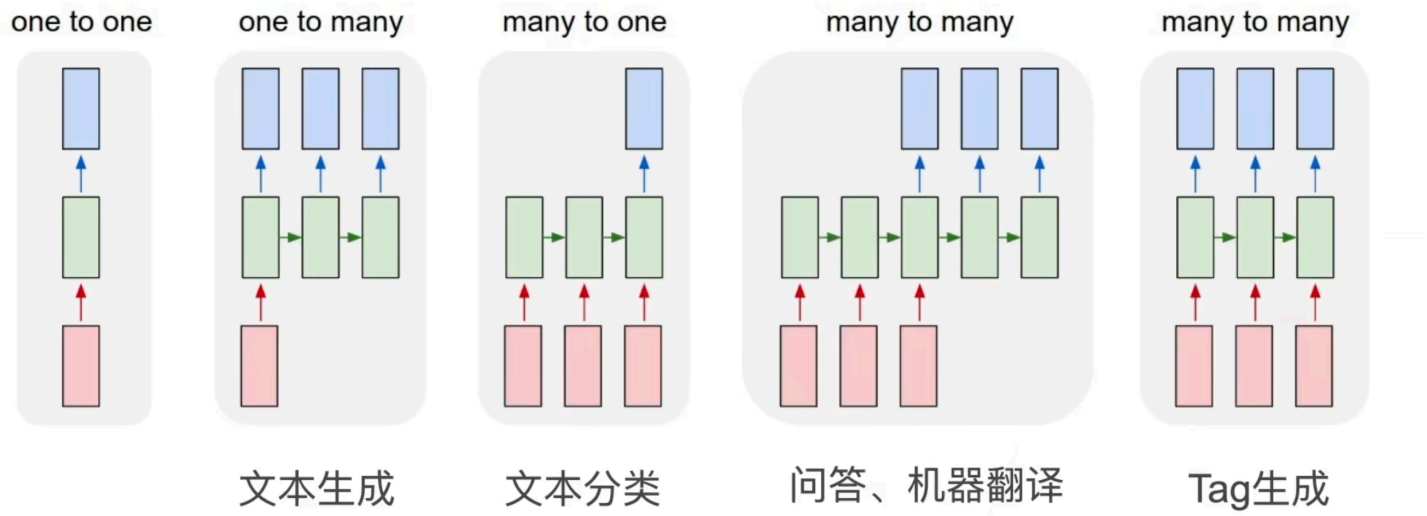
困惑度的定义为：测试集平均每个词的**概率倒数的指数**，数学上表达为：

$$\pi = \frac{1}{n}\Sigma_{i=1}^{n} - log\, p(x_i|x1, \ldots, x_{i-1}) \tag{4}$$

$$PP = exp(\pi) \tag{5}$$

直观解释：困惑度可以理解为模型在预测序列中下一个词时平均选择数，理想的完美模型困惑度应为1，这意味着模型总是能准确预测每个词。

应用



门控循环单元(GRU)

GRU是循环神经网络的一个变种，它使用两个门结构来控制隐藏状态的更新。这样做是因为在序列输入中，不是每个观察值都有相同的重要性，我们通常只需要关注几个关键点。

门的定义

门是与隐藏状态长度一样的向量，可以从输入数据和前一时刻的隐藏状态中学习得到。

- 更新门(Z)：关注机制，是否根据输入更新隐藏状态。
- 重置门(R)：遗忘机制，更新候选项时是否考虑前一隐藏状态

$$\begin{cases} R_t = \sigma(X_t W_{xr} + H_{t-1} W_{ht} + b_r) \\ Z_t = \sigma(X_t W_{xz} + H_{t-1} W_{hz} + b_z) \end{cases} \tag{6}$$

隐状态更新

与**RNN**不同，**GRU**的隐状态不直接进行更新，而是增加了一个候选隐状态。候选隐状态的更新方式与**RNN**类似，但是使用了**重置门**R_t来控制前一时刻隐状态对当前候选隐状态的影响能力。

- 如果$R_t = 1$，那么这个候选隐状态则与**RNN**没有区别。
- 如果$R_t = 0$，那么候选隐状态将不再考虑前一隐藏状态的影响，起到遗忘作用，这时模型与**MLP**没有区别。

$$\tilde{H}_t = tanh\left(X_t W_{xh} + (R_t \odot H_{t-1}) W_{hh} + b_h\right) \tag{7}$$

(其中⊙表示逐元素乘积)

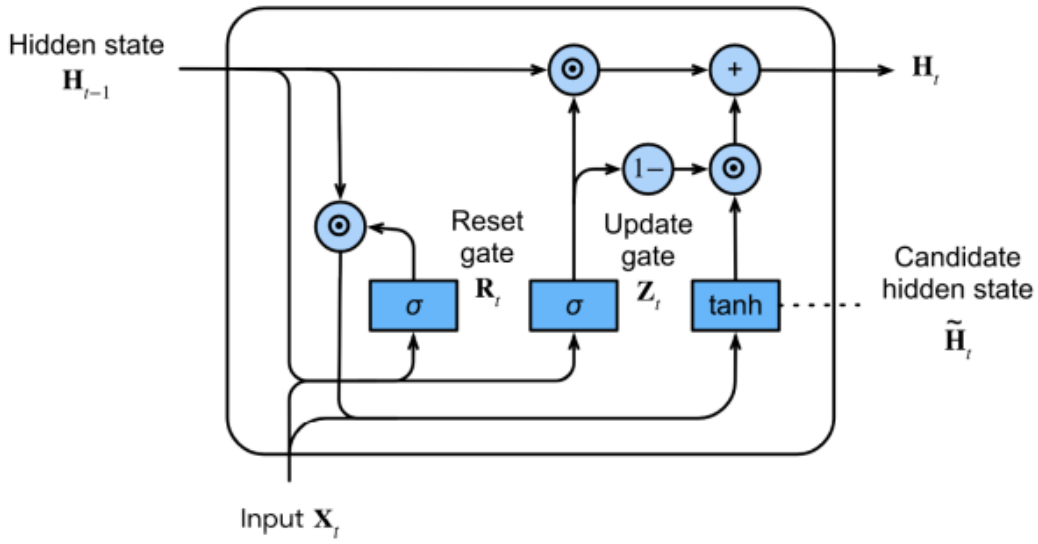
GRU的隐状态更新使用**更新门**Z_t来权衡上一时刻的隐状态与这一时刻的候选隐状态。

- 如果$Z_t = 0$，则完全更新，候选隐状态变为当前隐状态。
- 如果$Z_t = 1$，则完全不更新，当前隐状态直接沿用上一时刻的隐状态。

$$H_t = Z_t \odot H_{t-1} + (1 - Z_t) \odot \tilde{H}_t \tag{8}$$

总结

$$\boldsymbol{R}_t = \sigma(\boldsymbol{X}_t \boldsymbol{W}_{xr} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hr} + \boldsymbol{b}_r),$$
$$\boldsymbol{Z}_t = \sigma(\boldsymbol{X}_t \boldsymbol{W}_{xz} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hz} + \boldsymbol{b}_z)$$
$$\tilde{\boldsymbol{H}}_t = \tanh(\boldsymbol{X}_t \boldsymbol{W}_{xh} + \left(\boldsymbol{R}_t \odot \boldsymbol{H}_{t-1}\right) \boldsymbol{W}_{hh} + \boldsymbol{b}_h)$$
$$\boldsymbol{H}_t = \boldsymbol{Z}_t \odot \boldsymbol{H}_{t-1} + (1 - \boldsymbol{Z}_t) \odot \tilde{\boldsymbol{H}}_t$$



长短期记忆网络(LSTM)

LSTM与**GRU**类似，都是由门控制，但不同是的**LSTM**引入了**记忆单元**的概念。**记忆单元**可以理解为**隐状态**的一种特殊形式，他们拥有相同的形状，目的是为了记录更多的信息。另外，**LSTM**拥有三个门结构，分别为：

- **忘记门**(F)：用于决定是否重置记忆单元内容。
- **输入门**(I)：用于决定是否读取输入数据。
- **输出门**(O)：用于从记忆单元输出隐藏状态。

$$\begin{cases} F_t = \sigma(X_t W_{xf} + H_{t-1} W_{hf} + b_f) \\ I_t = \sigma(X_t W_{xi} + H_{t-1} W_{hi} + b_i) \\ O_t = \sigma(X_t W_{xo} + H_{t-1} W_{ho} + b_o) \end{cases} \tag{9}$$

记忆单元

记忆单元相当于在前一时刻隐状态和当前时刻隐状态之间加了一层，它更新由**候选隐藏单元**，**输入门**和**遗忘门**共同控制。

候选记忆单元从当前输入以及上一时刻输出获取：

$$\tilde{C}_t = tanh(X_t W_{xc} + H_{t-1} W_{hc} + b_c) \tag{10}$$

随后通过**输入门**和**遗忘门**进行更新

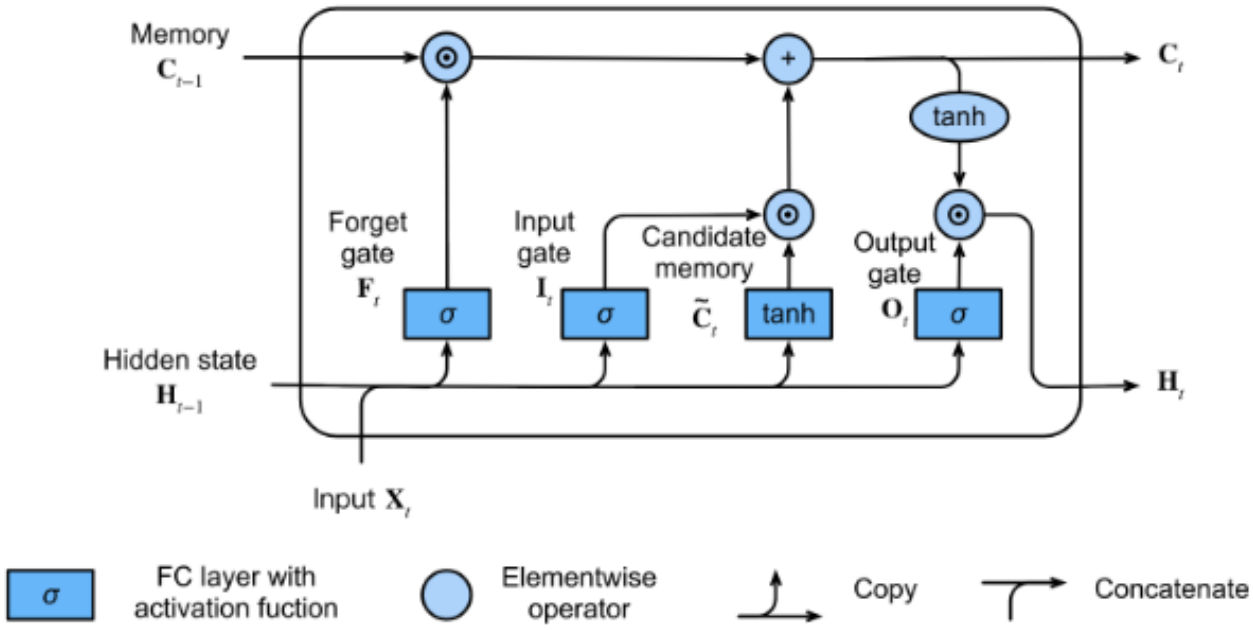
$$C_t = F_t \odot C_{t-1} + I_t \odot \tilde{C}_t\$ \tag{11}$$

隐状态

LSTM的隐状态由记忆单元通过输出门得到：

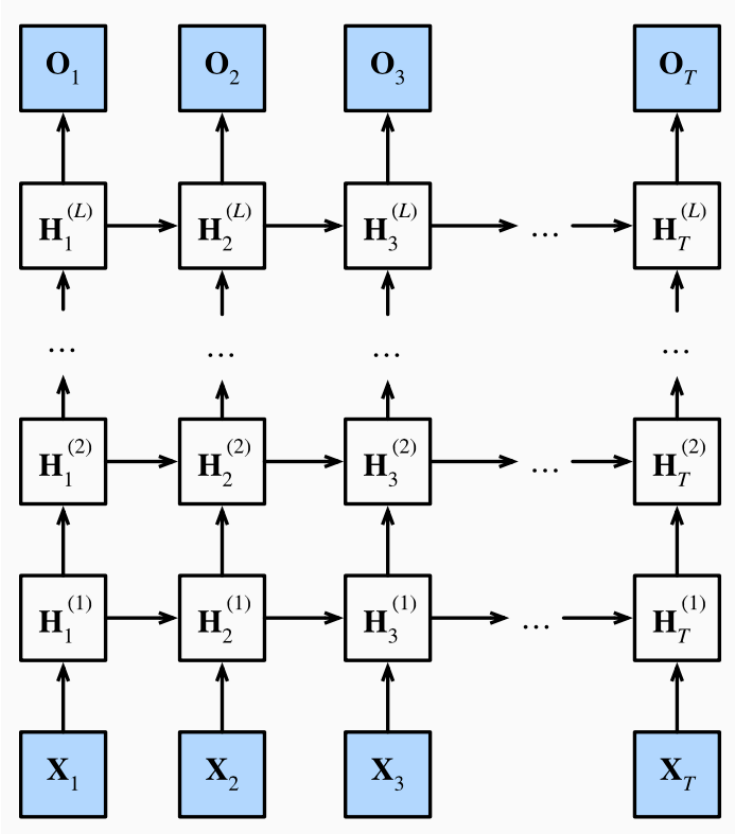
$$H_t = O_t \odot tanh(C_t) \tag{12}$$

流程图



深层循环神经网络

深层循环神经网络具有过个隐藏层，每层制作一点非线性，通过层数叠加获得更加非线性的模型。



公式

第1层的第t步隐状态是由第1层的第$t-1$步隐状态和第t步输入通过第1层函数f_1得到的：

$$H_t^1 = f_1(H_{t-1}^1, X_t) \tag{13}$$

第j层的第t步隐状态是由第j层$t-1$步隐状态和第$j-1$层的第t步隐状态通过第j层函数f_j得到的：

$$H_t^j = f_j(H_{t-1}^j, H_t^{j-1}) \tag{14}$$

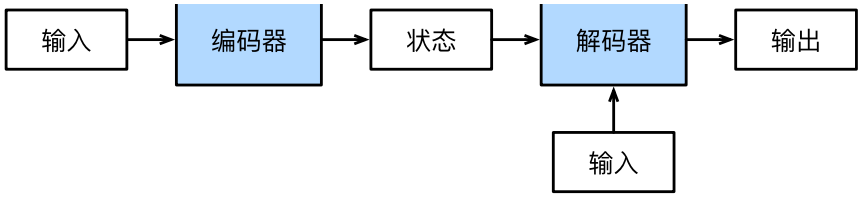
输出由最后一个隐藏层获得：

$$O_t = g(H_t^L) \tag{15}$$

编码器&解码器

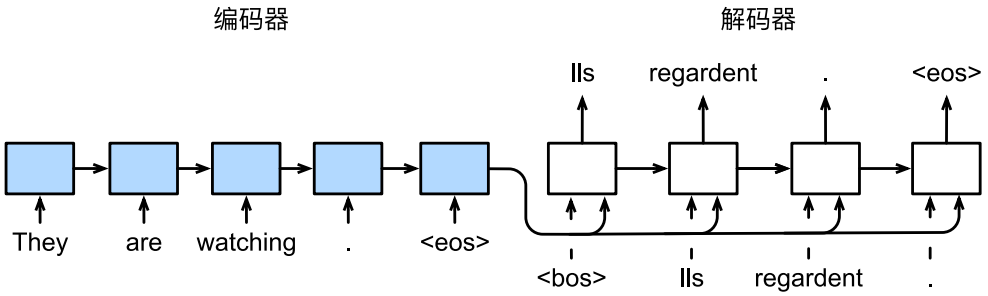
编码器解码器是一种深度学习模型架构，主要用于处理输入和输出为不同长度的序列任务。

- **编码器**将任意形状的输入数据编码，转化成模型可以识别的具有固定长度的编码状态（中间状态）。
- **解码器**将固定形状的编码状态映射到可变形状的输出数据。



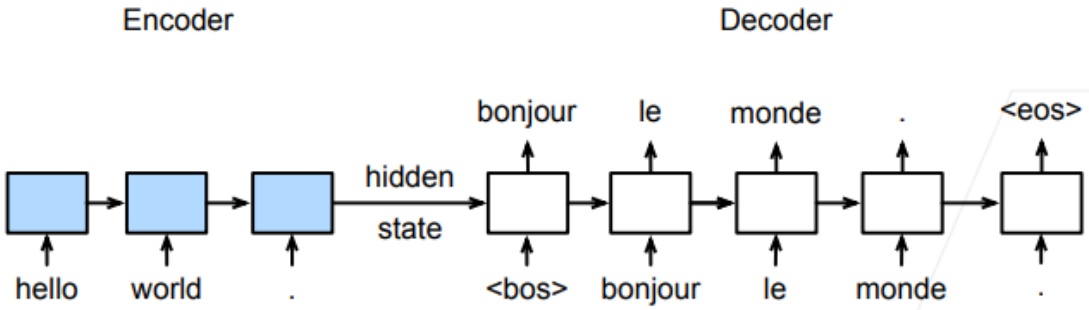
seq2seq模型

seq2seq模型是一种特别的编码器解码器架构，用于处理序列到序列的任务，例如机器翻译，聊天机器人等。

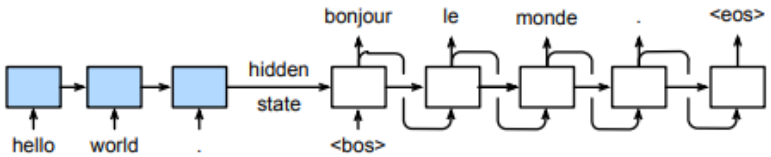


- **编码器**可以为**双向RNN**，因为输入为完整句子。
- **解码器**只能是单向，因为只能一个一个预测。
- **训练**时用目标句子做输入，而**推理**时用上一个输出做输入。

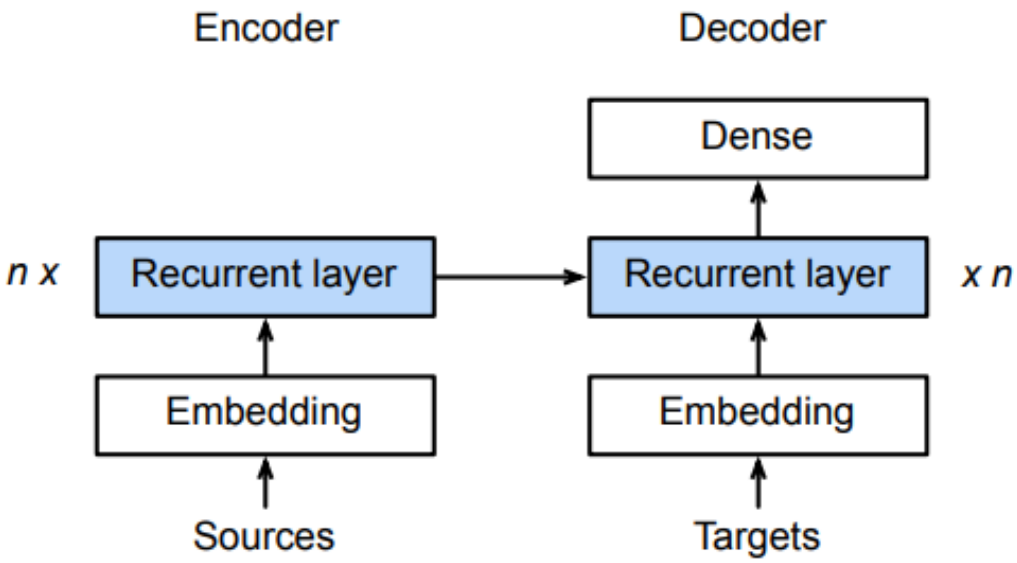
• 训练时解码器使用目标句子作为输入



推理：



- 编码器**最后时刻**隐状态为解码器**初始**隐状态。



BLEU

BLEU用于衡量生成序列的好坏。

$$BLEU = exp\left(min\left(0, 1 - \frac{len_{label}}{len_{pred}}\right)\right)\prod_{n=1}^{k} p_n^{1/2^n} \tag{16}$$

其中，p_n为预测中所有**n-gram**的精度。

- 标签序列**A B C D E F**和预测序列**A B B C D**有：

  $p_1 = 4/5, p_2 = 3/4, p_3 = 1/3, p_4 = 0.$

公式的前半部分**对过短的预测施加惩罚**，后半部分为预测精度。

束搜索

束搜索是一种基于**贪心搜索**的改进。与传统贪心搜索不同，束搜索每次不止考虑单一的最佳选项，而是保留前k个可能性最高的选项进行下一步生成。这些选项被称为**Beam**，数量被称为**Beam Width**。这样做的目的是为了找到总体概率更高的序列。

- $k = 1$时为贪心搜索。
- k越小搜索速度越快，但效果越差，k越大搜索速度越慢，但效果越好。
- 束搜索只在测试时使用。

生成的序列**评价标准**：

$$\frac{1}{L^\alpha}\log P(y_1,\ldots,y_L \mid \mathbf{c}) = \frac{1}{L^\alpha}\sum_{t'=1}^{\tilde{}} \log P(y_{t'} \mid y_1,\ldots,y_{t'-1},\mathbf{c}) \tag{17}$$

通常α取值为0.75，目的为给长序列更高的权重。