现代循环神经网络

门控循环单元 GRU

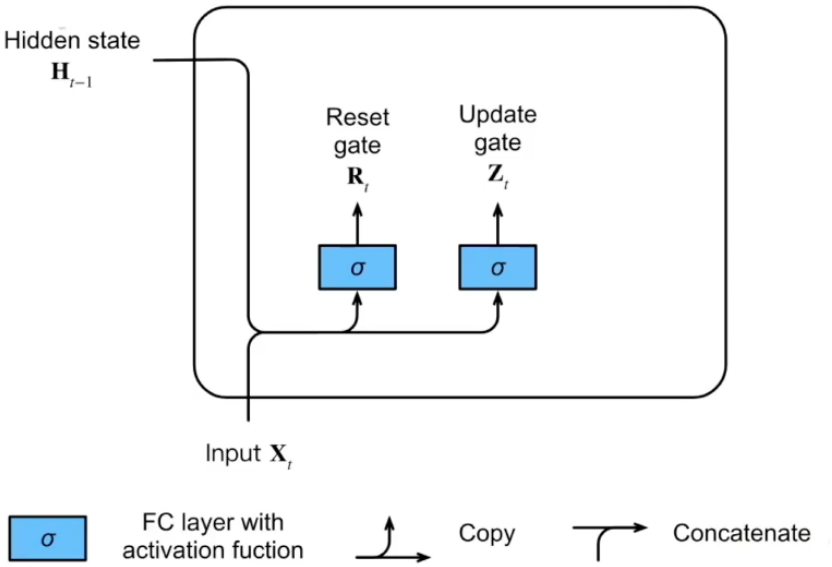
1. 引言 - 关注一个序列

- 不是每个观察值都是同等重要

- 想只记住相关的观察需要:
  a) 能关注的机制（更新门）
  b) 能遗忘的机制（重置门）

2. 具体结构

2.1 门



$$\boldsymbol{R}_t = \sigma(\boldsymbol{X}_t\boldsymbol{W}_{xr} + \boldsymbol{H}_{t-1}\boldsymbol{W}_{hr} + \boldsymbol{b}_r),$$
$$\boldsymbol{Z}_t = \sigma(\boldsymbol{X}_t\boldsymbol{W}_{xz} + \boldsymbol{H}_{t-1}\boldsymbol{W}_{hz} + \boldsymbol{b}_z)$$

其中，$\boldsymbol{R_t}$和$\boldsymbol{Z_t}$与$\boldsymbol{H_{t-1}}$形状一致

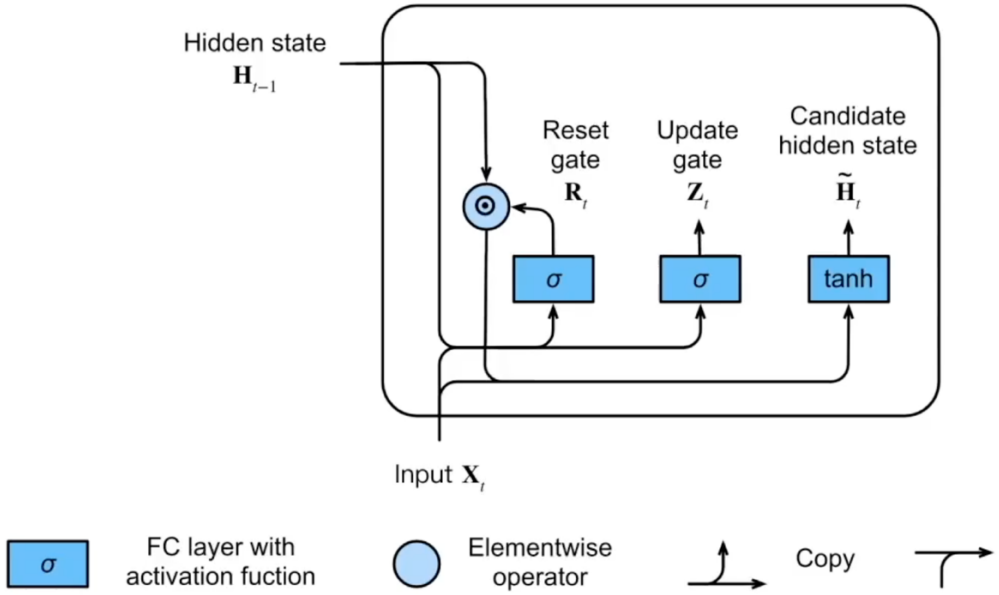
2.2 候隐藏状态

$$\tilde{\boldsymbol{H}}_t = \tanh(\boldsymbol{X}_t\boldsymbol{W}_{xh} + \left(\boldsymbol{R}_t \odot \boldsymbol{H}_{t-1}\right)\boldsymbol{W}_{hh} + \boldsymbol{b}_h)$$

其中，⊙表示**按元素乘法**

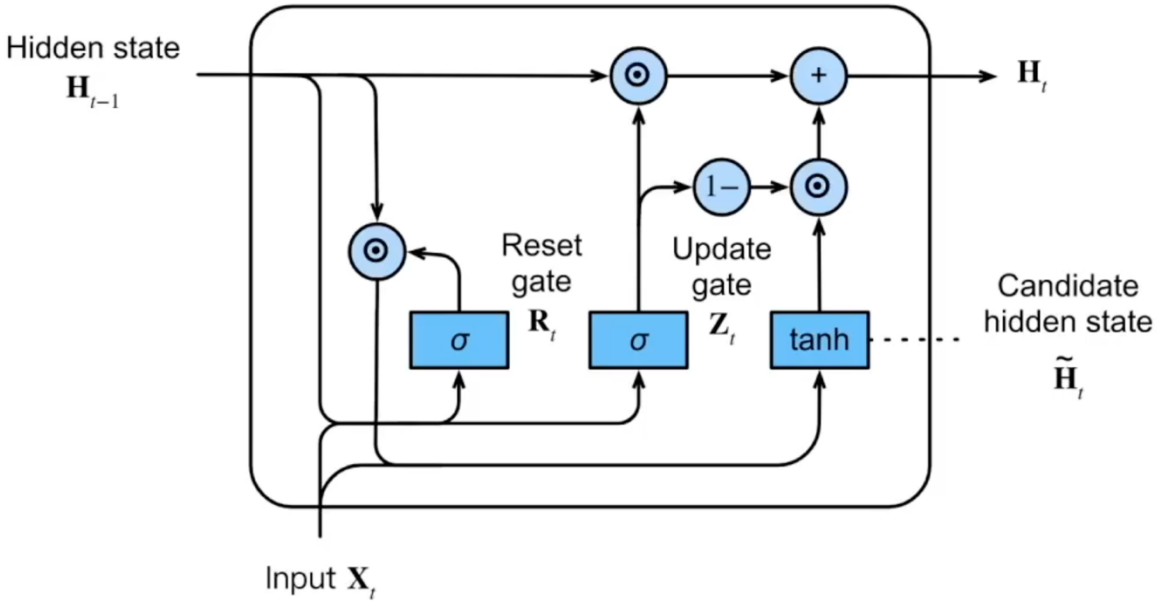
- 所谓**重置**:

  假设激活函数采用 SoftMax，则$\boldsymbol{R_t}$中元素值的表示范围在 0 ~ 1 之间，所以利用元素乘法可以将$\boldsymbol{H_{t-1}}$中元素进行置 0 操作，例如极端情况下，$\boldsymbol{R_t}$为一个全零矩阵。



2.3 隐状态

$$\boldsymbol{H}_t = \boldsymbol{Z}_t \odot \boldsymbol{H}_{t-1} + (1 - \boldsymbol{Z}_t) \odot \tilde{\boldsymbol{H}}_t$$



长短期记忆网络 LSTM

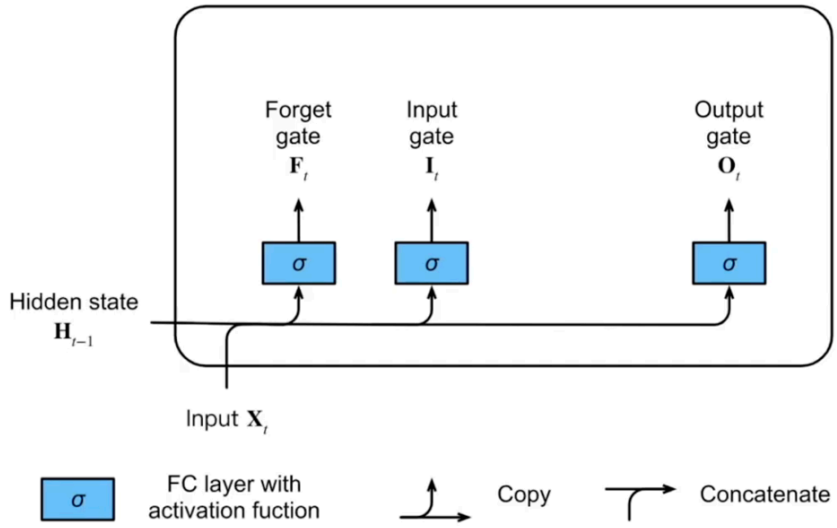
1. 引言

- 忘记门: 将值朝 0 减少
- 输入门: 决定是不是忽略掉输入数据
- 输出门: 决定是不是使用隐状态

2. 具体结构

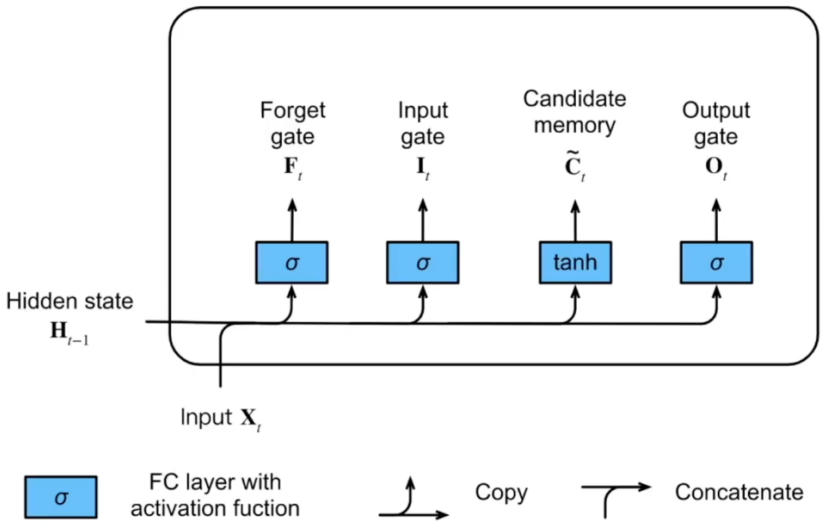
2.1 门

$$
\begin{aligned}
\boldsymbol{I}_t &= \sigma(\boldsymbol{X}_t \boldsymbol{W}_{xi} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hi} + \boldsymbol{b}_i) \\
\boldsymbol{F}_t &= \sigma(\boldsymbol{X}_t \boldsymbol{W}_{xf} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hf} + \boldsymbol{b}_f) \\
\boldsymbol{O}_t &= \sigma(\boldsymbol{X}_t \boldsymbol{W}_{xo} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{ho} + \boldsymbol{b}_o)
\end{aligned}
$$



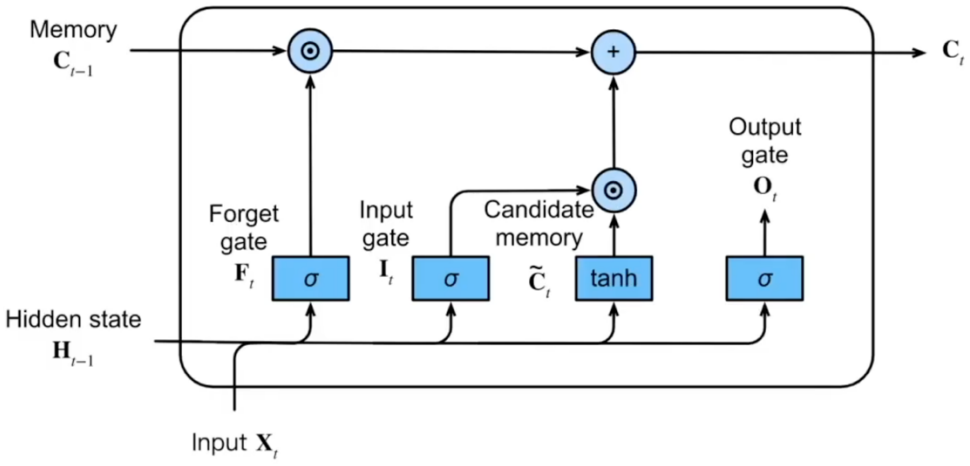
2.2 候选记忆单元

$$
\tilde{\boldsymbol{C}}_t = \tanh(\boldsymbol{X}_t \boldsymbol{W}_{xc} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hc} + \boldsymbol{b}_c)
$$



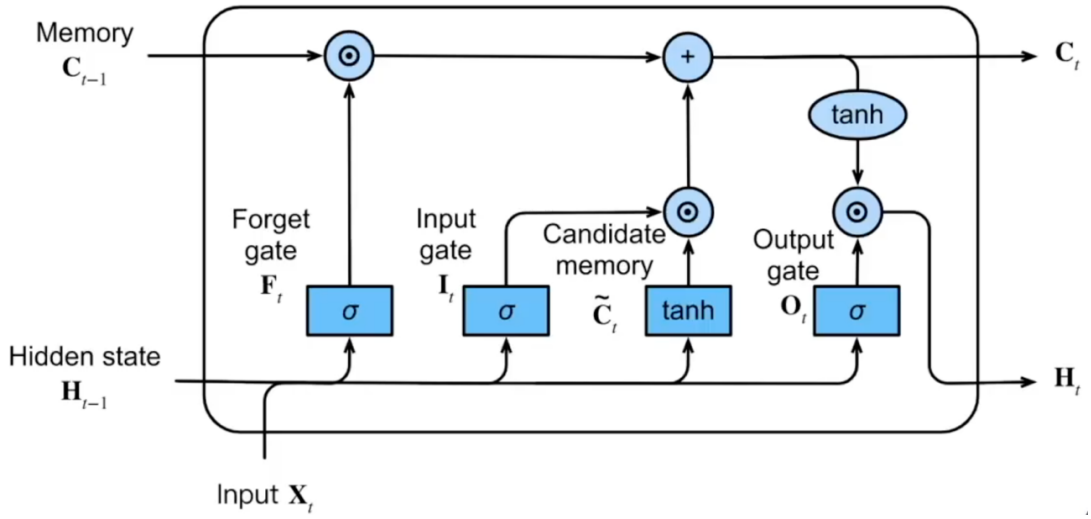
2.3 记忆单元

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



2.4 隐状态

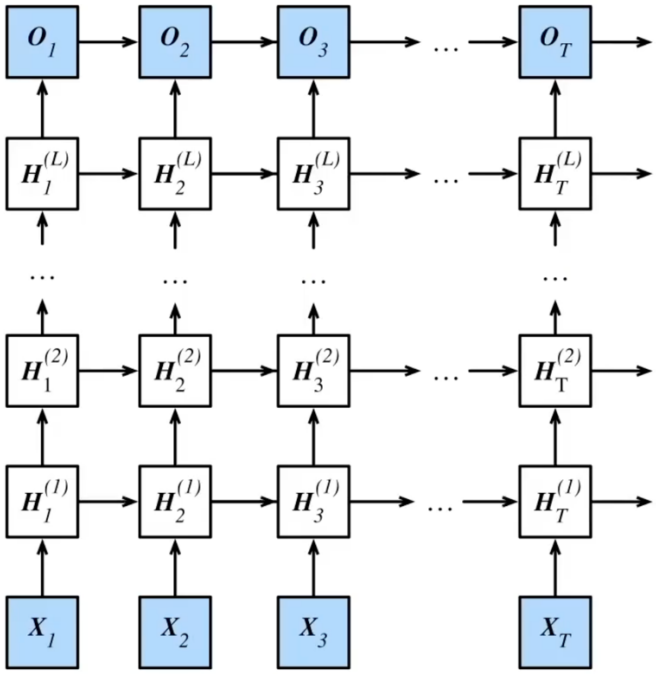
$$H_t = O_t \odot \tanh(C_t)$$



深度循环神经网络

- 方式：深度循环神经网使用更多的隐藏层来获得更多的非线性

- 浅 RNN
  - 输入
  - 隐层
  - 输出
- 深 RNN
  - 输入
  - 隐层
  - 隐层
  - …
  - 输出



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

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

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

双向循环神经网络

1. 引言 –‘未来很重要’

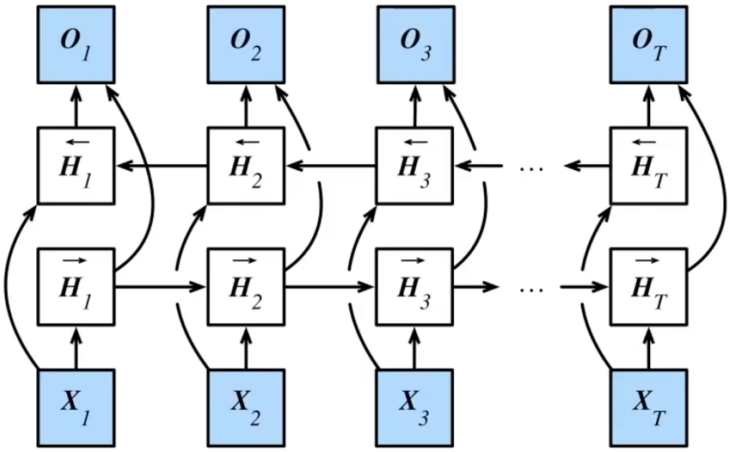
```
I am _____
I am _____ very hungry,
I am _____ very hungry, I could eat half a pig
```

```
I am happy.
I am not   very hungry,
I am very  very hungry, I could eat half a pig
```

- 取决于过去和未来的上下文，可以填写很不一样的词
- 目前为止，RNN 只看过去
- 在填空的时候，也看未来

2. 双向 RNN

- 一个前向的 RNN 隐层
- 一个反向的 RNN 隐层
- 合并两个隐状态得到输出



$$\overrightarrow{\mathbf{H}}_t = \phi(\mathbf{X}_t \mathbf{W}_{xh}^{(f)} + \overrightarrow{\mathbf{H}}_{t-1} \mathbf{W}_{hh}^{(f)} + \mathbf{b}_h^{(f)}),$$

$$\overleftarrow{\mathbf{H}}_t = \phi(\mathbf{X}_t \mathbf{W}_{xh}^{(b)} + \overleftarrow{\mathbf{H}}_{t+1} \mathbf{W}_{hh}^{(b)} + \mathbf{b}_h^{(b)}),$$

$$\mathbf{H}_t = [\overrightarrow{\mathbf{H}}_t, \overleftarrow{\mathbf{H}}_t]$$

$$\mathbf{O}_t = \mathbf{H}_t \mathbf{W}_{hq} + \mathbf{b}_q$$

3. 推理

- 训练：

- 推理：

- 很难进行推理，因为既需要看到之前的信息，也需要看到之后的信息。
- 双向 RNN 主要是对句子进行特征提取，例如翻译/改写，能看见句子的全部信息，或者是语音输入，听见一个完整的句子。

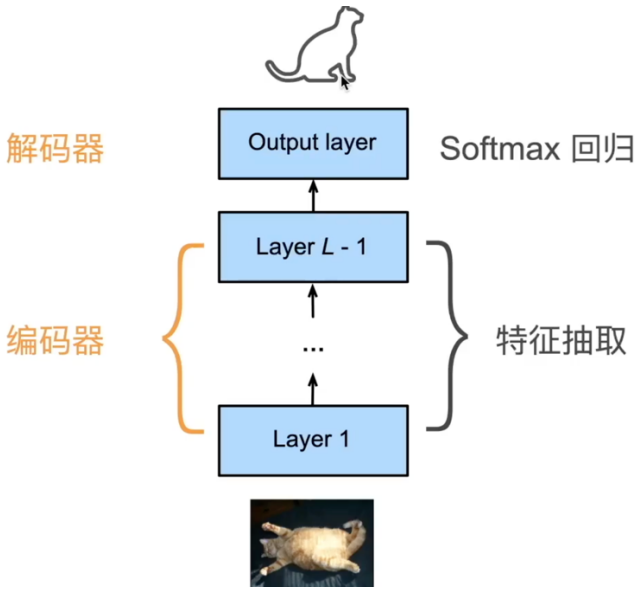
4. 总结

- 双向循环神经网络通过反向更新的隐藏层来利用方向时间信息
- 通常用来对序列抽取特征、填空，而不是预测未来

编码器-解码器结构

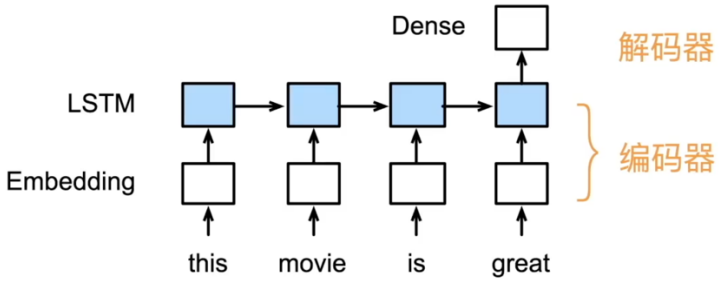
1. 重新考察 CNN

- 编码器：将输入编程成中间表达形式（特征）
- 解码器：将中间表达解码成输出



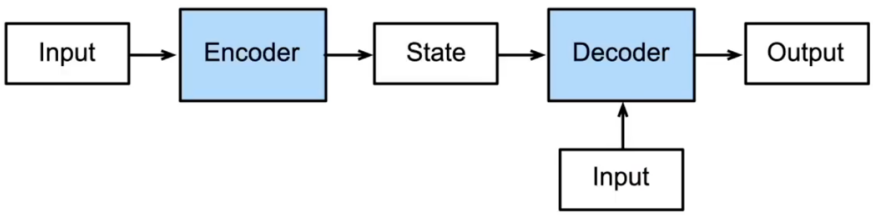
2. 重新考察 RNN

- 编码器：将文本表示成向量
- 解码器：向量表示输出



3. 编码器-解码器架构

- 一个模型被分为两块
  a) 编码器处理输入
  b) 解码器生成输出

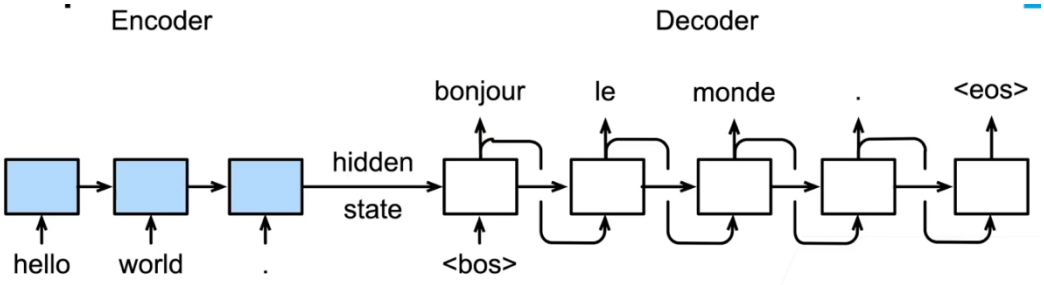


序列到序列学习 seq2seq

1. 机器翻译

- 给定一个源语言的句子，自动翻译成目标语言
- 这两个句子可以有不同的长度

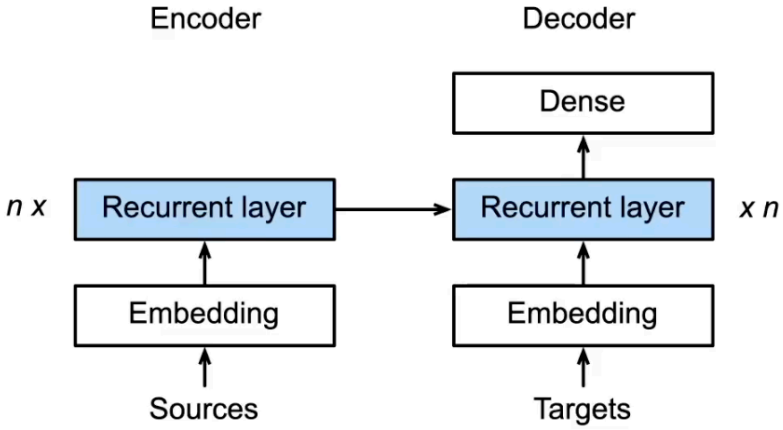
2. seq2seq



- 编码器是一个 RNN，读取输入句子**（可以是双向）**
- 解码器使用另外一个 RNN 来输出

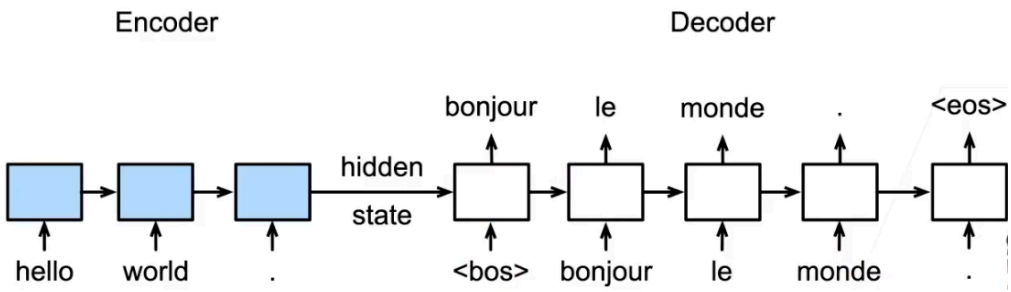
2.1 细节

- 编码器是没有输出的 RNN
- 编码器最后时间步的隐状态用作解码器的初始隐状态

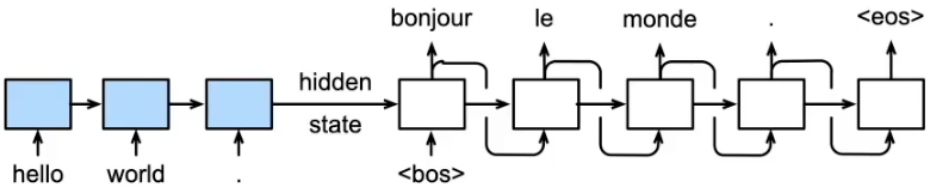


2.2 训练

- 训练时解码器使用目标句子作为输入



2.3 推理



3. 衡量生成序列好坏的 BLEU

- p_n 是预测中所有 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
    $p_1 = 4/5,\ p_2 = 3/4,\ p_3 = 1/3,\ p_4 = 0$
- BLEU定义

$$\exp\left(\min\left(0,1-\frac{\text{len}_{\text{label}}}{\text{len}_{\text{pred}}}\right)\right)\prod_{n=1}^{k}p_n^{1/2^n}$$

惩罚过短的预测　　长匹配有高权重

4. 总结

- Seq2seq 从一个句子生成另一个句子
- 编码器和解码器都是 RNN
- 将编码器最后时间隐状态来初始解码器隐状态来完成信息传递
- 常用 BLEU 来衡量生成序列的好坏

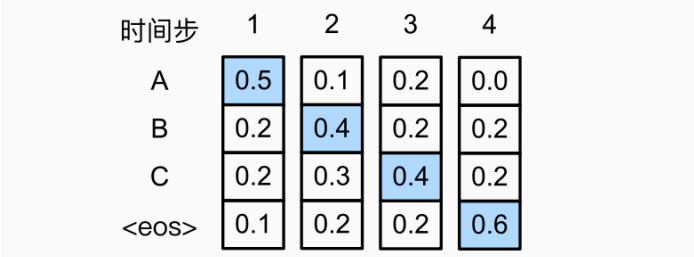
束搜索

1. 贪心搜索

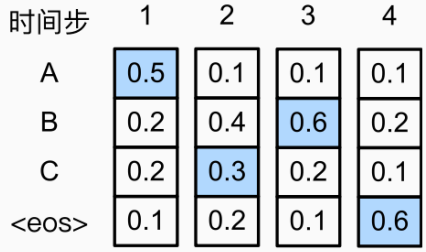
- 在 seq2seq 中我们使用了贪心搜索来预测序列

  将当前时刻预测概率最大的词输出

- 但贪心很可能不是最优的



在每个时间步，贪心搜索选择具有最高条件概率的词元



在时间步2，选择具有第二高条件概率的词元“C”（而非最高条件概率的词元）

上图 1 中，预测输出序列“A”、“B”、“C”和“<eos>”。 这个输出序列的条件概率是

$$0.5 \times 0.4 \times 0.4 \times 0.6 = 0.048$$

上图 2 中，输出序列 “A”、“C”、“B”和“<eos>”的条件概率为

$$0.5 \times 0.3 \times 0.6 \times 0.6 = 0.054$$

说明贪心搜索存在的问题如下:

现实中，最优序列（optimal sequence）应该是最大化值的输出序列，这是基于输入序列生成输出序列的条件概率。 然而，贪心搜索无法保证得到最优序列。

2. 穷举搜索

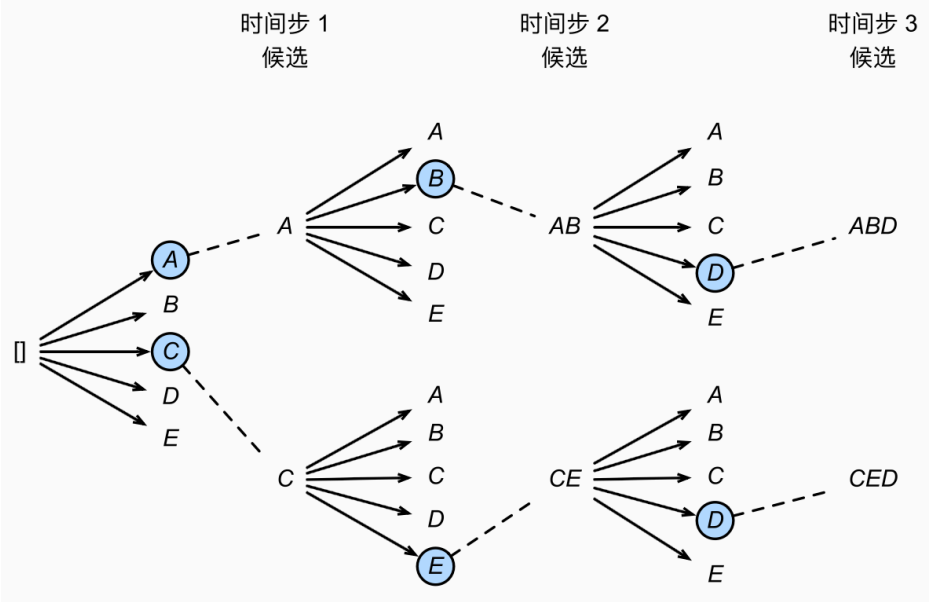
- 最优算法: 对所有可能的序列，计算它的概率，然后选取最好的那个
- 如果输出字典的大小为n，序列最长为T，那么需要考察n^T个序列

$$n = 1000, \quad T = 10, \quad n^T = 10^{10}$$

在计算上不可行

3. 束搜索

- 保存最好的k个选项
- 在每个时刻，对每个候选新加一项（n种可能），在kn个选项中选出最好的k个



3.1 复杂度

- 时间复杂度 $O(knT)$
  - $k = 5, \quad n = 10000, \quad T = 10: \quad knT = 5 \times 10^5$
- 每个候选的最终分数是:

$$\frac{1}{L^{\alpha}} logp(y_1, \ldots, y_L) = \frac{1}{L^{\alpha}} \sum_{t'}^{L} (logp(y_{t'} | y_1, \ldots, y_{t'-1}))$$

4. 总结

- 束搜索在每次搜索时保存 k 个最好的候选
  - $k = 1$ 时是贪心搜索
  - $k = n$ 时是穷举搜索