23 编码器解码器，Seq2seq模型，束搜索

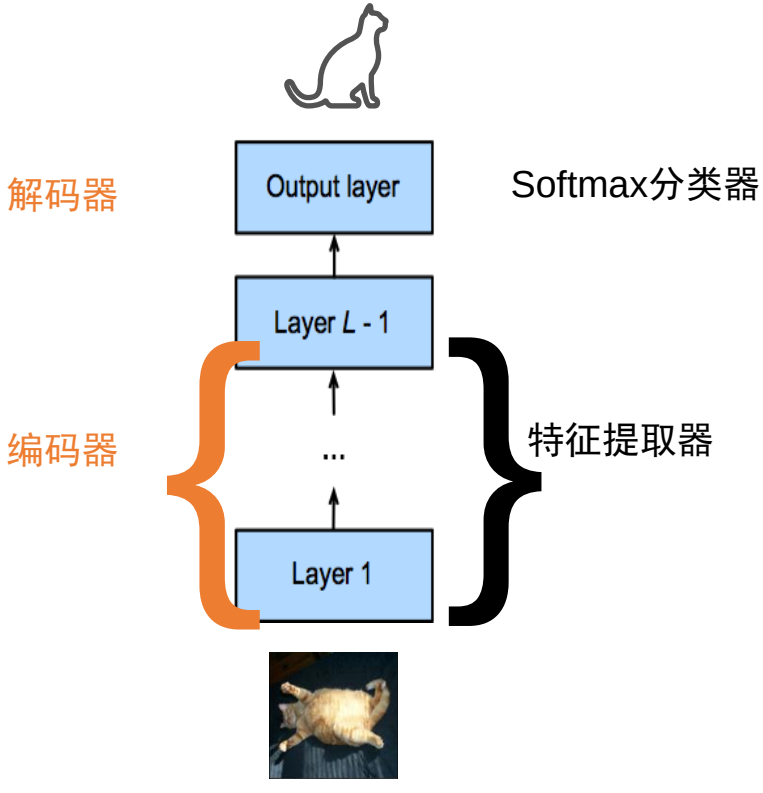
概要

- 编码器 - 解码器 (Encoder - Decoder)
- Seq2seq 模型
- 束搜索（Beam Search）
  - 贪婪搜索（Greedy Search）
  - 穷举搜索（Exhaustive Search）
  - 束搜索

编码器 - 解码器
(Encoder - Decoder)

重新思考CNN

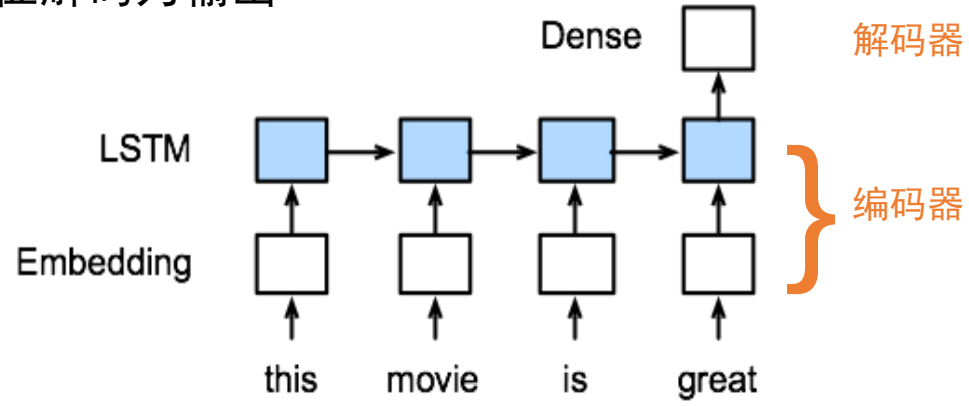
- 编码器(Encoder)
  - 将输入编码为中间表示（特征）
- 解码器(Decoder)
  - 将特征解码为输出



重新思考RNN

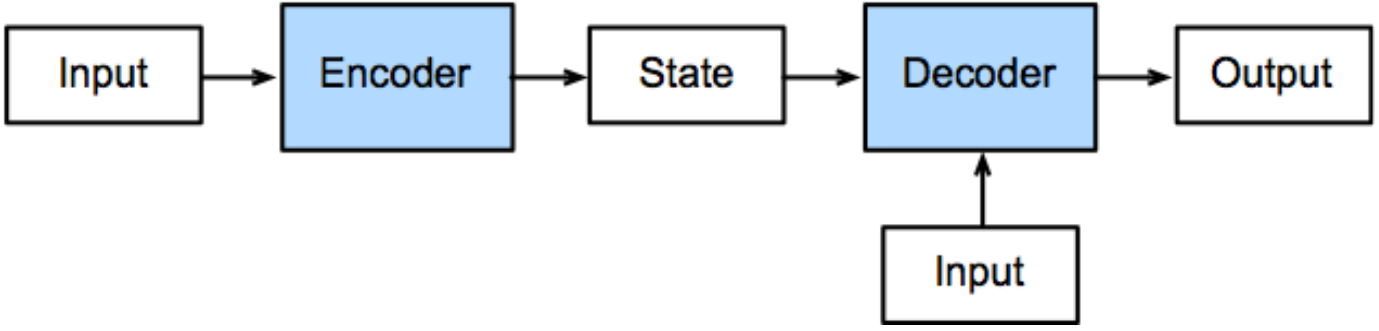
- 编码器 (Encoder)
  - 将输入编码为中间表示（特征）

- 解码器(Decoder)
  - 将特征解码为输出



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 Class

- 只指定长度可变的序列作为编码器的输入

```
1. class Encoder(nn.Module):
2.     """编码器-解码器架构的基本编码器接口"""

3.     def __init__(self, **kwargs):
4.         super(Encoder, self).__init__(**kwargs)

5.     def forward(self, X, *args):
6.         raise NotImplementedError
```

解码器Class

- 使用编码器输出和任何其他信息创建状态
- 解码器在每个时间步都会将输入（如前一时间步生成的词元）和编码后的状态映射成当前时间步的输出词元

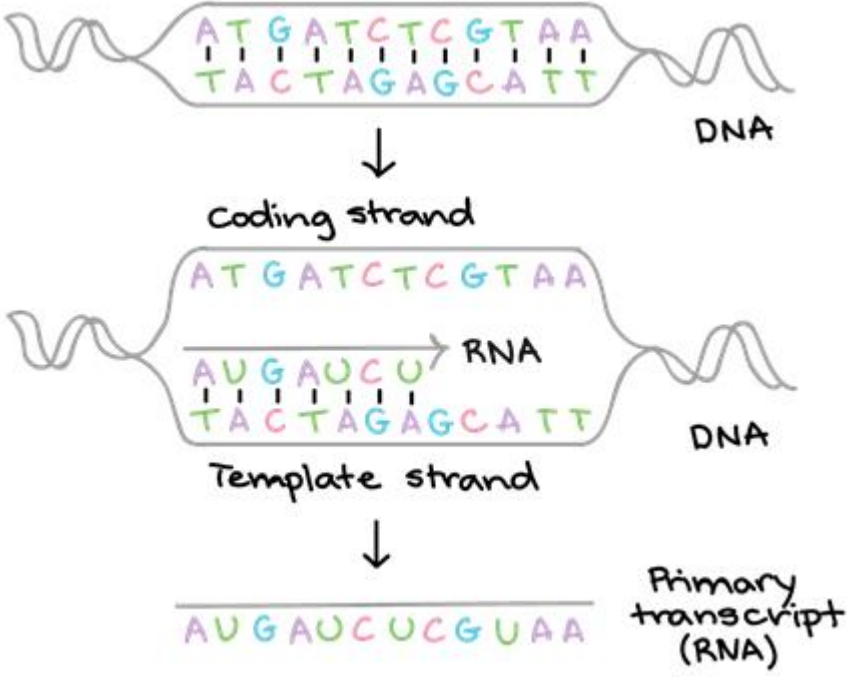
```
1. class Decoder(nn.Module):
2.     """编码器-解码器架构的基本解码器接口"""
3.     def __init__(self, **kwargs):
4.         super(Decoder, self).__init__(**kwargs)

5.     def init_state(self, enc_outputs, *args):
6.         raise NotImplementedError

7.     def forward(self, X, state):
8.         raise NotImplementedError
```

编码器-解码器 Class

- 在前向传播中，编码器的输出用于生成编码状态， 这个状态又被解码器作为其输入的一部分
- “状态” 也可以使用具有状态的神经网络来实现

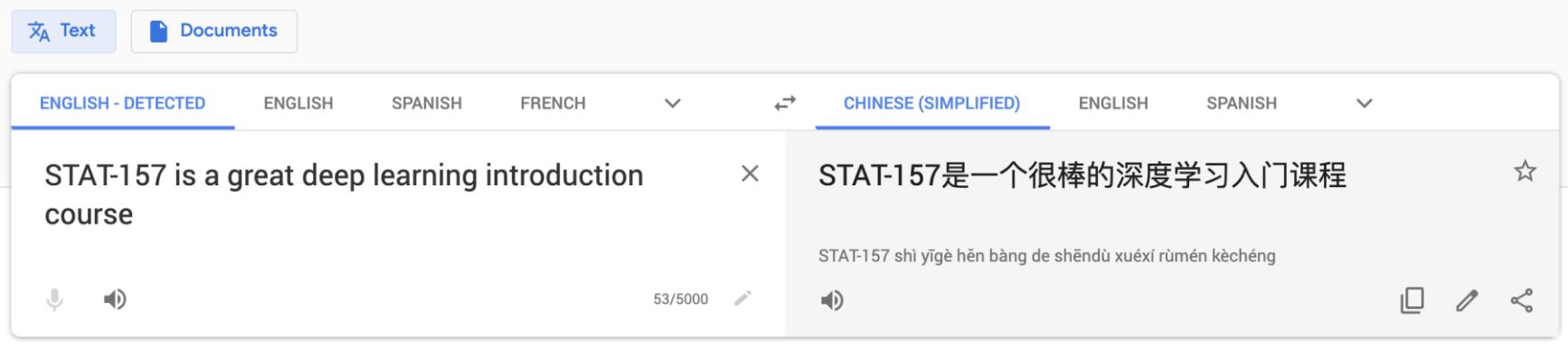
```
1.  class EncoderDecoder(nn.Module):
2.      """编码器-解码器架构的基类"""
3.      def __init__(self, encoder, decoder, **kwargs):
4.          super(EncoderDecoder, self).__init__(**kwargs)
5.          self.encoder = encoder
6.          self.decoder = decoder

7.      def forward(self, enc_X, dec_X, *args):
8.          enc_outputs = self.encoder(enc_X, *args)
9.          dec_state = self.decoder.init_state(enc_outputs, *args)
10.         return self.decoder(dec_X, dec_state)
```

Seq2seq模型


ATGATCTCGTAA
TACTAGAGCATT
DNA
coding strand
ATGATCTCGTAA
RNA
AUGAUCU
TACTAGAGCATT
DNA
Template strand
AUGAUCUCGUAA
Primary transcript (RNA)

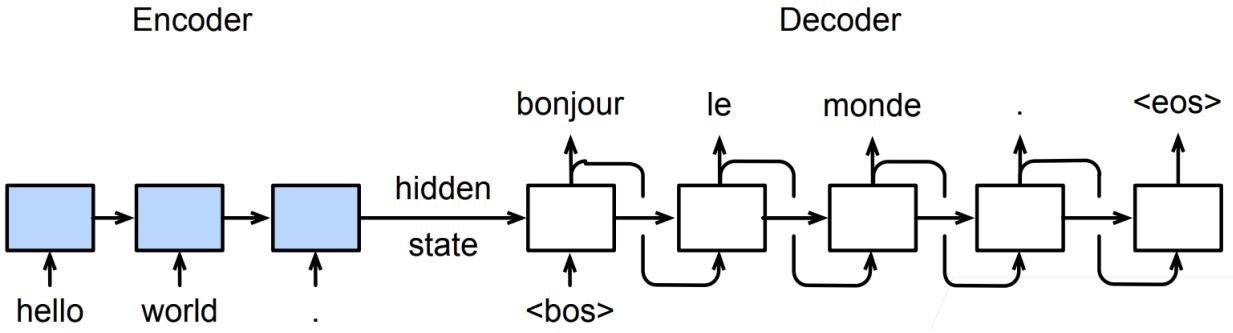
机器翻译

- 给定源语言中的句子，翻译成目标语言
  - 输入输出两个序列可以具有不同的长度



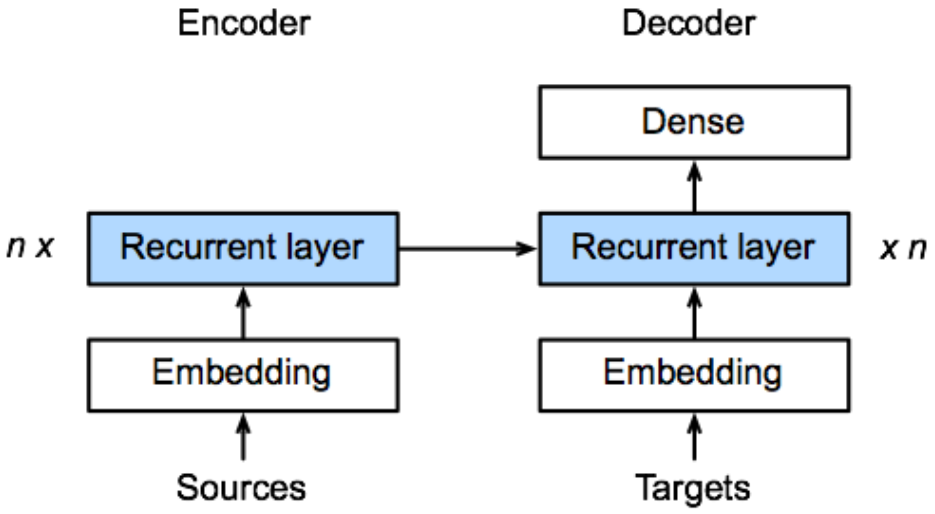
Seq2seq模型

- 编码器：读取输入序列的RNN
  - 使用长度可变的序列作为输入， 将其转换为固定形状（长度）的隐状态
  - 可以双向【解码器不能双向】
- 解码器：另一个RNN生成输出
  - 为了连续生成输出序列的词元， 解码器基于输入序列的编码信息和输出序列已经看见的或者生成的词元来预测下一个词元
  - 从最后一个时刻生成的hidden state开始



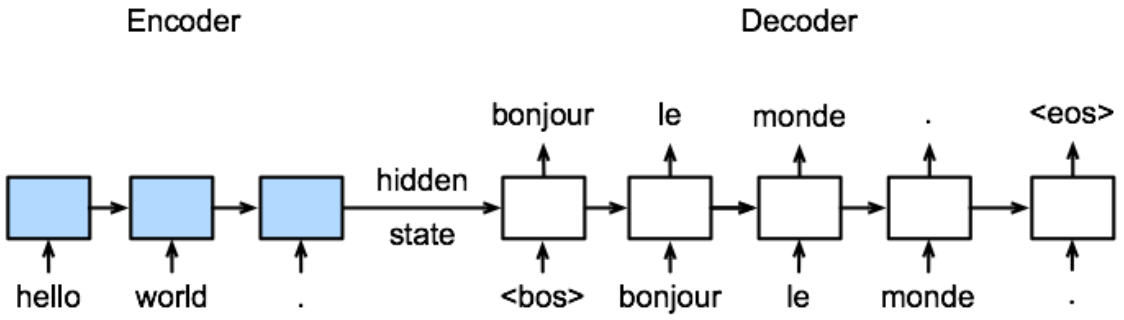
Seq2seq 模型

- 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

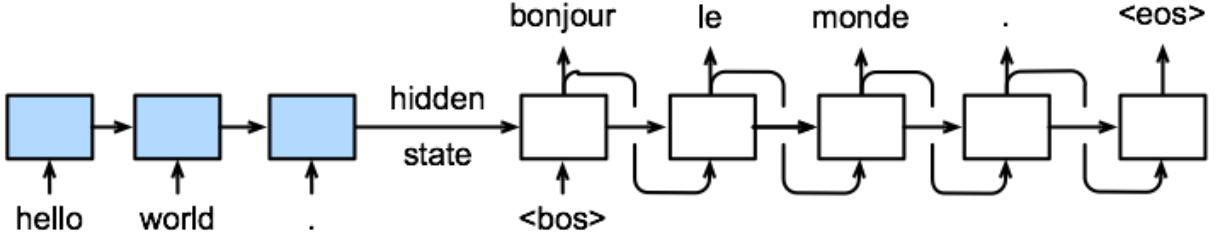


机器翻译训练

- 在训练期间，解码器（Decoder）用目标语言句子作为输入
  - 推理时，输入为上一时刻的输出



Predict:



衡量生成序列的好坏的BLEU

- p_n 是预测中所有 n-gram 的精度
  - 标签序列 $ABCDEF$ 和预测序列 $ABBCD$, 有
  - $p_1 = 4/5, p_2 = 3/4, p_3 = 1/3, p_4 = 0$
- BLEU定义（越大越好）

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

  - 惩罚过短的预测
  - 长匹配有高权重

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

代码...

束搜索Beam Search

贪婪搜索

- 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
  - 当前最优不一定导致整体最优

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:
$0.5\times0.3\times0.6\times0.6=0.054$

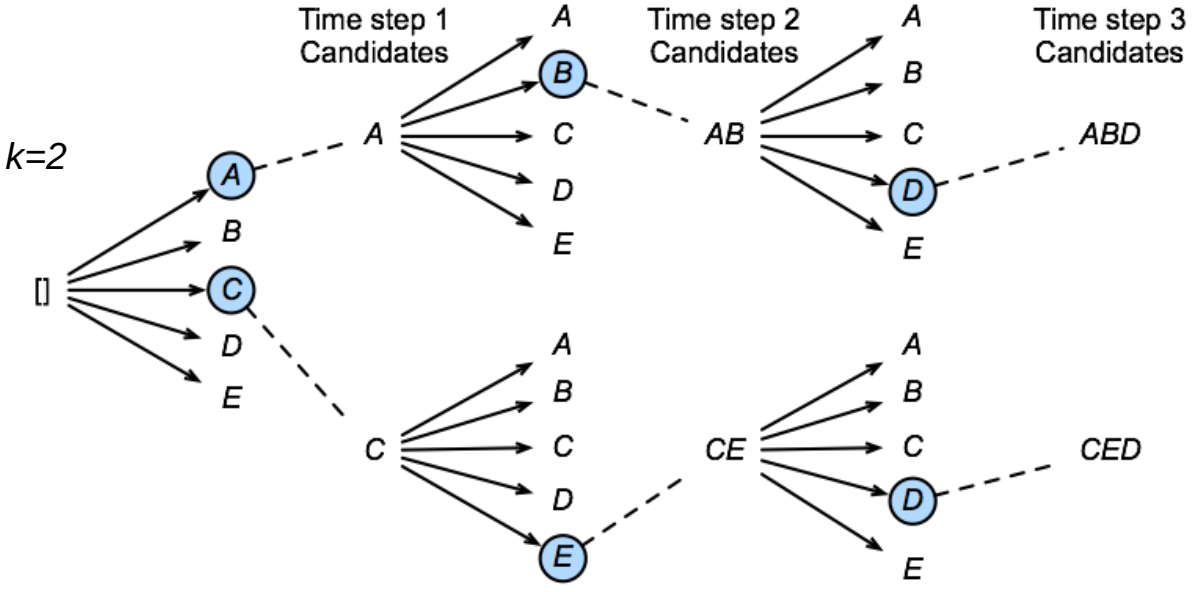
时间步	1	2	3	4
A	0.5	0.1	0.1	0.1
B	0.2	0.4	0.6	0.2
C	0.2	0.3	0.2	0.1
<eos>	0.1	0.2	0.1	0.6

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，那么需要检查 n^T 序列
  - 计算上不可行
  - $n = 10000, T = 10: n^T = 10^{40}$

束搜索

- 每次都保留最好的 k个候选
- 向候选项添加一项（n种可能），在kn序列选出最好的k个



束搜索

- 时间复杂度 $O(knT)$
  - $k = 5,\ n = 10000,\ T = 10$: $knT = 5 \times 10^5$

- 每个候选的最终得分是

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

  - 常用值$\alpha = 0.75$
  - 鼓励长句子

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