

### 第三章 问答摘要与推理-Seq2Seq (一)

## HCT NLP Week 3

### 问答摘要与推理 Seq2Seq (一)

#### Outline

- Encoder-Decoder结构
- Attention机制
- 模型Layer、Model构建
- Seq2Seq训练

#### Outline

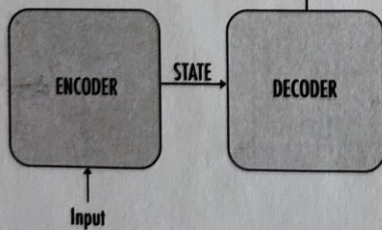
- Encoder-Decoder结构
- Attention机制
- 模型Layer、Model构建
- Seq2Seq训练

## Seq2Seq

### Encoder-Decoder结构

Sequence to sequence was first introduced by Google in 2014

Seq2Seq应用场景:



1. Speech Recognition 语音识别

2. Machine Language Translation 机器翻译

3. Name entity/Subject extraction 实体识别/主题识别

4. Relation Classification 关系分类

5. Path Query Answering QA问题

6. Speech Generation 语音生成问题

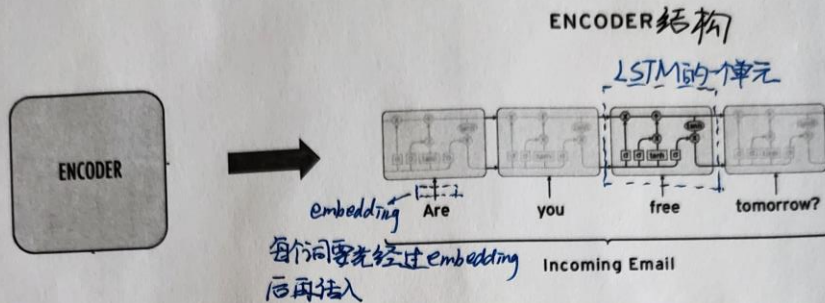
7. Chatbot 聊天机器人

8. Text Summarization 文本摘要

9. Product Sales Forecasting 产品销售预测

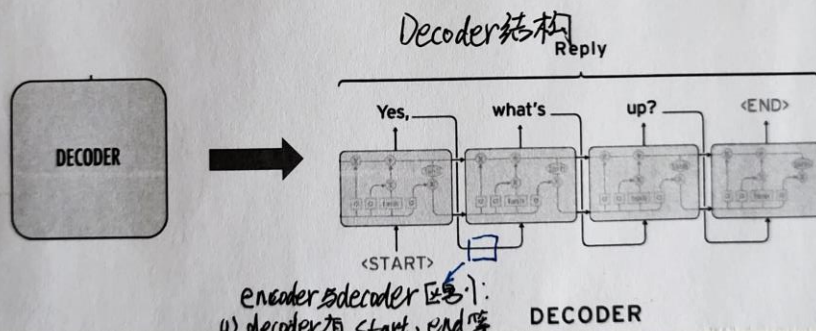
## Seq2Seq

### Encoder-Decoder结构



## Seq2Seq

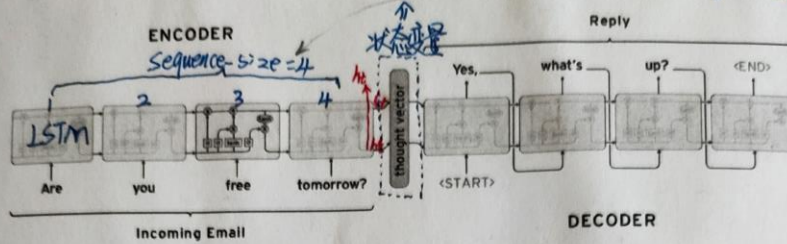
### Encoder-Decoder结构





## Seq2Seq

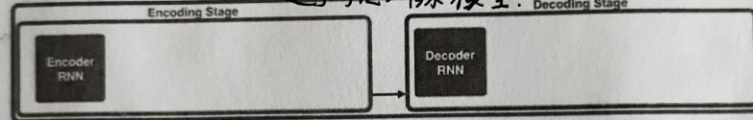
### Encoder-Decoder结构



## Seq2Seq

### Encoder-Decoder结构

Neural Machine Translation  
SEQUENCE TO SEQUENCE MODEL



Encoding时传递的为状态变量。

注:动态图中decoder没有画输入

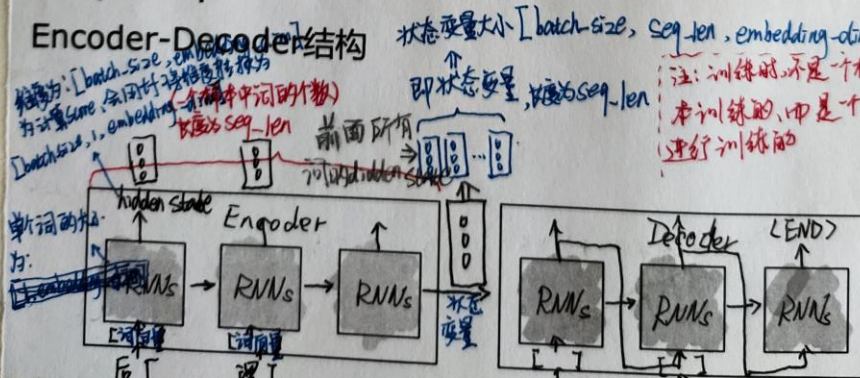
Encoder-Decoder结构动态图

网址

(source: Jay. Alamar, 2018)

## Seq2Seq

### Encoder-Decoder结构



假设一次喂给神经网络 batch-size 个样本, 每个样本中词的数量为 seq-len, 则一次 Encoder 的维度为 [batch-size, seq-len, embedding-dim]

① 题没有词表: 今天/天气/很好/....., 词表中总共有1万个词, 记  $V_{\text{voc}}=1\text{万}$   
 ② 没有样本: [今天, 天气] ③ 设样本总数为  $L_{\text{samp}}, D_{\text{v}} | L_{\text{samp}} \neq L_{\text{voc}}$   
 [今天, 很好] ④ 设  $L_{\text{samp}}=2\text{万}$ , 取  $\text{batch-size}=64$ , (即训练时每次  
 给神经网络的数据为64)  
 ⑤ 设词向量维度为300维, 记  $\text{embedding-dim}=300, D_{\text{v}}$   
 ⑥ 设样本中间词: [天气, 今天]  
 ⑦ 设样本中间词: [天气, 很好]  
 ⑧  $\text{seq-len}=2$

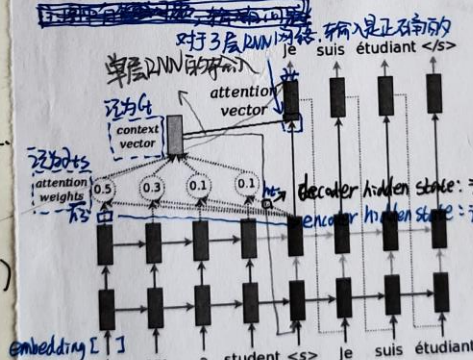
结论: ① embedding大小为: [1万, 300]

② Encoder-decoder中, 一次训练Encoder的大小为  $[\text{batch-size}, \text{seq-len}, \text{embedding-dim}]$   
 ③ 中间的状态变量大小为  $[\text{batch-size}, \text{seq-len}, \text{embedding-dim}]$

## Outline

- Encoder-Decoder结构
- Attention机制
- 模型Layer、Model构建
- Seq2Seq训练

## Attention机制 (看下一张图, 再看此图)



$$C_t = \sum_s a_{ts} \cdot \bar{h}_s$$

如  $= 0.5 \bar{h}_1 + 0.3 \bar{h}_2 + \dots$

$$d_t = f(C_t, h_t) = \tanh(W_c [C_t, h_t])$$

注:

① 摘要通常只放一层神经网络

② 给定一个词, context vector 都要重新计算

③ decoder 和 attention 是同时计算的

④ 在一次训练中, 先 encoder 再 decoder

Score 的计算方式有2种:

注: 其中  $W, W_1, W_2$  为神经网络的权

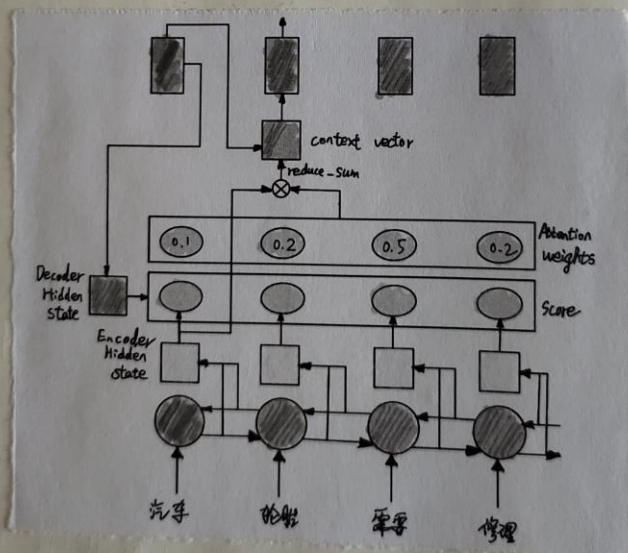
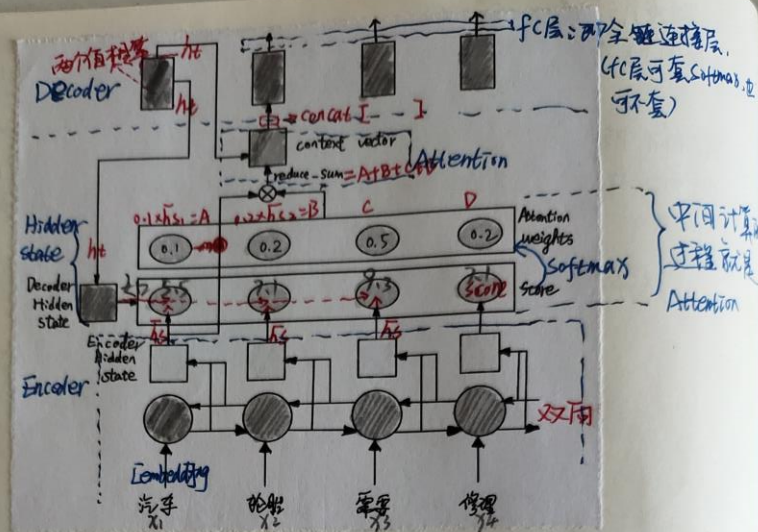
重矩阵, 不是 embedding

$$\text{Score} = \begin{cases} h_t^T \cdot W \cdot \bar{h}_s & (\text{Luong's, multiplicative style}) \\ V_a^T \cdot \tanh(W_1 h_t + W_2 \bar{h}_s) & (\text{Bahdanau's, additive style}) \end{cases}$$

$$d_{ts} = \frac{e^{\text{score}}}{\sum_{s=1}^S e^{\text{score}_s}}$$

Softmax





注: 摘要抽取和翻译问题最大的不同: 前者encoder, decoder词向量共享, 后者不共享, 后者有两个数据集, 两个词向量。

## Outline

- Encoder-Decoder结构
- Attention机制
- 模型Layer、Model构建
- Seq2Seq训练

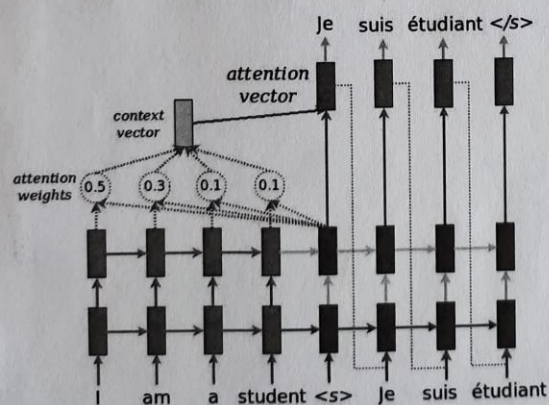
模型Layer、Model构建

代码见jupyter

## Outline

- Encoder-Decoder结构
- Attention机制
- 模型Layer、Model构建
- Seq2Seq训练

## Seq2Seq训练



How is it trained?

## Seq2Seq训练

cross-entropy loss 交叉熵

交叉熵描述了两个概率分布之间的距离

注: 神经网络输出不一定是概率分布。往往使用softmax将输出变为概率分布

假设一次Decoder ~~输入~~ 输出为  $z = [z_1, z_2, z_3]$ ,

假设:  $z_1$  输出 3  
 $z_2$  输出 1  
 $z_3$  输出 -3

$$\text{Softmax} \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}} = 0.88$$

$$= 0.12$$

$$= 0.00$$

Softmax产生的概率分布  
 (概率一般不为0)

$S(y)$  预测值

0.88
0.12
0.00

$$D(S, L) = -\sum_i L_i \log(S_i)$$

交叉熵

$L$  对应的真实标签

1
0
0

即使概率为0  
 也不会出现计算错误

随着Decoder 序列长度的增加, Loss函数取交叉熵均值。

$J = -\frac{1}{N} \sum_i y_i \log \hat{y}_i$  其中  $\hat{y}_i$  为预测值,  $y_i$  为真实标签,  $N$  为decoder序列长度

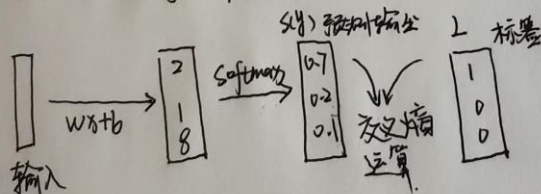
即:  $L = -\frac{1}{N} \sum_i y_i \log \hat{y}_i$

对于二分类问题:

$$L = -\frac{1}{N} \sum_i [y \log \hat{y} + (1-y) \log (1-\hat{y})]$$

$y$  的取值为0或者1.

整个训练过程可如下表示:





seq2seq训练

解决摘要问题中生成的词没在文中出现过，导致句子很别扭的问题。

其他训练技巧：“先验知识”

decoder  $\Rightarrow \text{softmax}(y) \Rightarrow P_i = \frac{e^{y_i}}{\sum_j e^{y_j}}$

在decoder过程中加入先验知识，使得选取词为文中出现过的词

先验知识引入：设词典大小为  $|V|$ ，为 0/1 向量表示先验知识，构建张量

量  $x = (x_1, x_2, \dots, x_n)$

$x_i = 1$ ：表示在文中出现过  
 $x_i = 0$ ：表示没在文中出现过

在decoder时通过shift引入先验知识：

$\hat{y} = S @ x + t = (Sx_1 + t_1, Sx_2 + t_2, \dots, Sx_n + t_n)$

学习参数(训练参数)，通过神经网络进行训练

$\frac{\hat{y} + y}{2} = y \Rightarrow \text{softmax} \rightarrow P_i = \frac{e^{y_i}}{\sum_j e^{y_j}}$

最终输出

注：引入先验知识即使取出的词更多的都来自源文中

## Seq2Seq训练技巧

2 Teacher Forcing 即将前一步输出的真实结果作为下一步的输入

RNN模型是用前一步的输出作为输入

Teacher Forcing方法能够解决收敛速度慢和不稳定的问题

## Extensions to Teacher Forcing

普通训练：

$\langle \text{START} \rangle$  奔驰汽车的方向盘  $\langle \text{END} \rangle$

↑ 预测 ↑  
 $\langle \text{START} \rangle$  成的  $\langle \text{START} \rangle$  的？

第三个输入本来应该为“汽车”，错误预测成了的，再将“START”的传入进行预测，导致错的越来越多（不收敛）



Teacher forcing:

\*  $\langle \text{START} \rangle$  奔驰 汽车 的 .....  $\langle \text{END} \rangle$   
          ↑          ↑          ↑  
           $\langle \text{START} \rangle$   $\langle \text{START} \rangle$  奔驰  $\langle \text{START} \rangle$  奔驰汽车.

Teacher forcing 即不管预测是否对错, 都将上一步的正确值代入下一步的输入.

注: 预测时采用 teacher forcing 会带来 exposure bias 问题 (即如果前面错了, 后面会跟着错).

## Seq2Seq 训练

预测联合概率:

Exposure Bias

$P(y|x) = P(y_1|x)P(y_2|y_1)P(y_3|y_1, y_2) \dots P(y_n|y_1, \dots, y_{n-1})$

在训练过程中产生的

解决 Exposure Bias:

~~在训练过程中解决:~~

训练  $\rightarrow$  Inference (推理)

① 在推理过程中 (测试) 解决, 即传入 2 个或多个概率最大的代入下一步

(2) ① Scheduled Sampling (前面几步给真实值, 后面几步给预测值)

② RL: 强化学习 (Reinforcement Learning)

③ 数据重采样 (类似 MASK 操作)

④ 对抗训练 (梯度惩罚)

附：

(1) OpenNMT: Open source ecosystem for neural machine translation and neural sequence learning: <https://github.com/OpenNMT>

(2) Encoder、Decoder、Attention 层解说博文：  
<https://blog.csdn.net/zimiao552147572/article/details/105893842>

(3) TensorFlow 官方 attention 文档“基于注意力的神经机器翻译”：  
[https://www.tensorflow.org/tutorials/text/nmt\\_with\\_attention](https://www.tensorflow.org/tutorials/text/nmt_with_attention)

(4) Google Cloab:  
[https://colab.research.google.com/github/tensorflow/docs-l10n/blob/master/site/zh-cn/tutorials/text/nmt\\_with\\_attention.ipynb](https://colab.research.google.com/github/tensorflow/docs-l10n/blob/master/site/zh-cn/tutorials/text/nmt_with_attention.ipynb)

注：从如下接口进入 Google Cloab

The screenshot shows the TensorFlow website interface. The top navigation bar includes links for '安装' (Install), '学习' (Learn), 'API', '资源' (Resources), '社区' (Community), and '选择 TensorFlow 的原因' (Reasons to choose TensorFlow). The main content area is titled 'TensorFlow Core' and features a sidebar with navigation links for '概览' (Overview), '教程' (Tutorials), '指南' (Guides), and 'TF 1'. The '教程' (Tutorials) section is expanded, showing a list of tutorials under '面向专家的快速入门' (Quickstart for experts). The '文本' (Text) section is highlighted, and the tutorial '基于注意力的神经机器翻译' (Neural Machine Translation with Attention) is selected. The main content area displays the tutorial title, a '在 Google Colab 运行' (Run on Google Colab) button, and a '在 GitHub 上查看源' (View source on GitHub) button. A note states: '★ Note: 我们的 TensorFlow 社区翻译了这些文档。因为社区翻译是英文文档。如果您有改进此翻译的建议，请提交 pull request 到 docs-zh-cn@tensorflow.org Google Group。' (★ Note: Our TensorFlow community translated these documents. Because community translations are English documents. If you have suggestions for improving this translation, please submit a pull request to docs-zh-cn@tensorflow.org Google Group.)