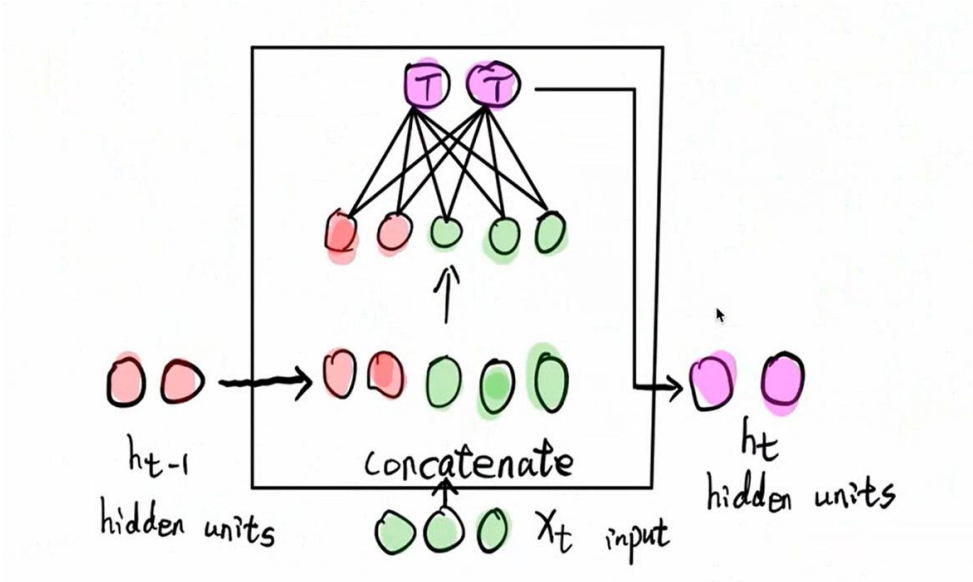
基于 seq2seq 架构

一、seq2seq 模型的基本架构

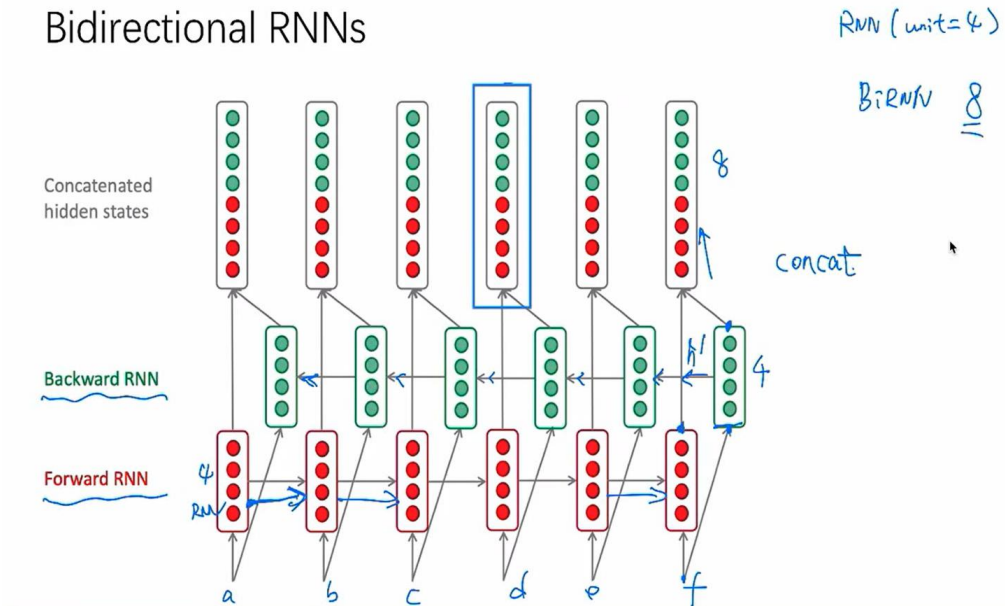
1、rnn 的基本原理

rnn 是一个带有时序的神经网络，代码基本实现：将当前输入 x_t 与上一时刻的 h_t-1 进行 concat,紧接着进行全连接得到 h_t。



rnn 原理图

相对一般的神经网络，rnn 可以更好的参数共享，而一般情况下，大家比较喜欢用双向 Rnn，双向 rnn 能从正、反两个方面去考虑，提取特征较丰富；注意：双向 rnn 的输出=2*hidden_size，具体结构如下：



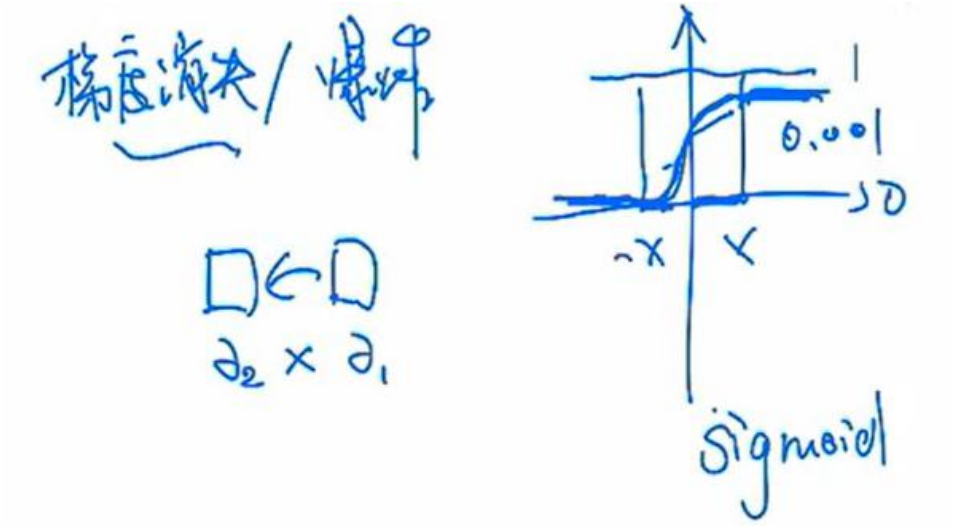
bi_rnn 结构图

此外，机器翻译最初用到多层 rnn，一般 2~4 效果最优。

2、LSTM 的基本原理

rnn 在遇到比较长的文本时候，就会出现梯度消失/爆炸的情况。因此后来就出现 lstm，想以此来解决梯度消失/爆炸的问题。

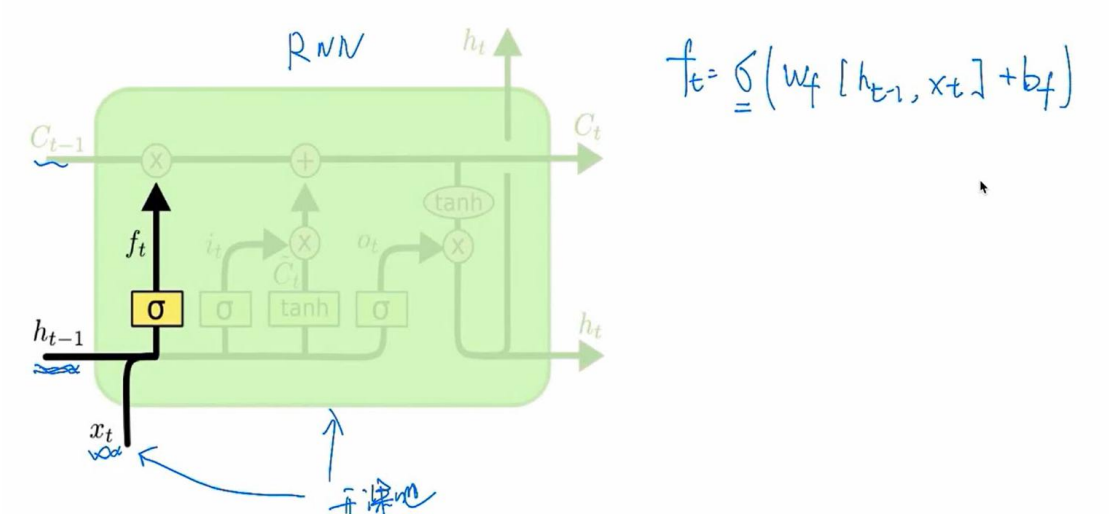
具体梯度消失/爆炸的原因的分析：rnn 结构采用矩阵乘+sigmoid/tanh(激活函数)，sigmoid 的取值在 0~1，根据下图分析，当 x 超过一定范围 x，线就趋于平滑，求梯度时就近似与 0，连乘时一连串趋于 0 的梯度相乘，梯度就消失了，参数就不能更新，这就是梯度消失。反之，当梯度一直大于 1，连乘之后就会出现梯度一直在变大，最终导致梯度爆炸。



Sigmoid 曲线图

后来，提出 lstm，缓解了这个问题，lstm 结构主要有三个门：遗忘门、输入门、输出门，具体分析如下：

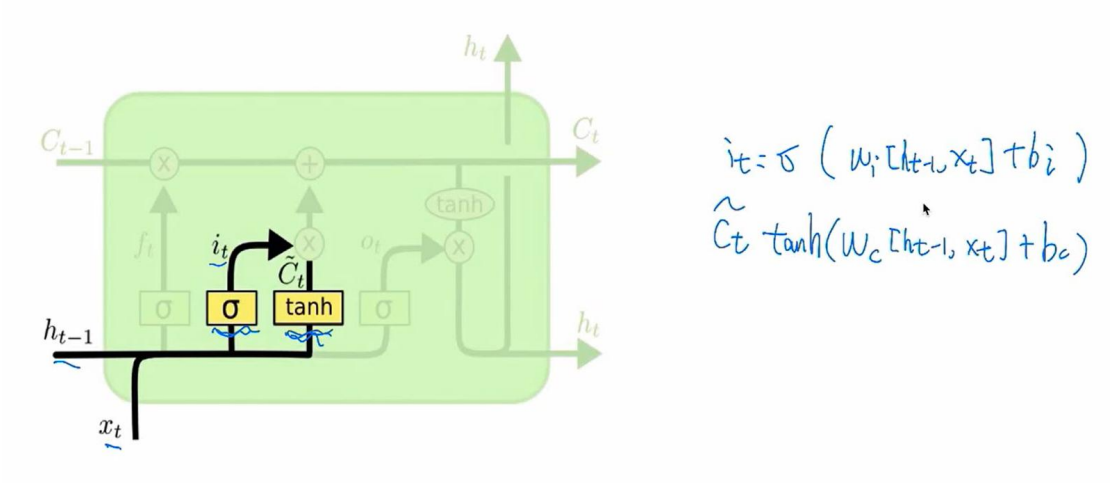
①遗忘门



遗忘门图

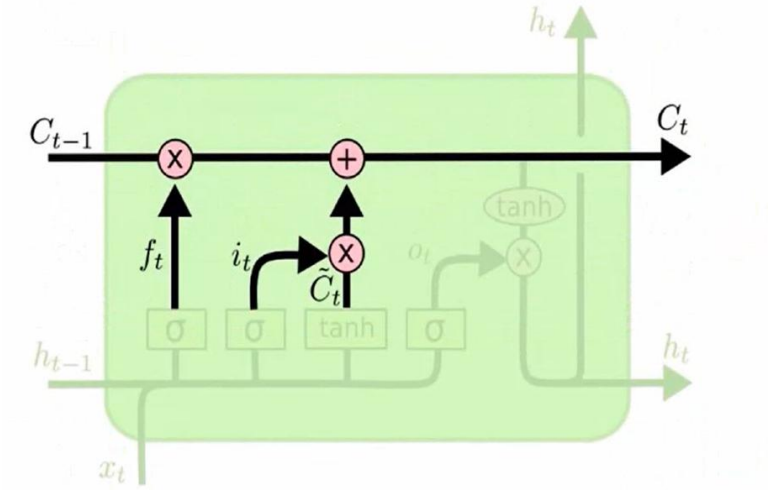
如图所示，遗忘门将上一时刻输出与当前 x_t 融合，使用 sigmoid 激活作为遗忘门，返回对应 c(t-1)的 shape 的值，值在 0-1 之间，表示对应的 c(t-1)的保留比率（因为比率都<=1,每一次与 c(t-1)相乘，都会有信息被遗忘），表示对 c(t-1)的过滤。

②输入门



输入门图

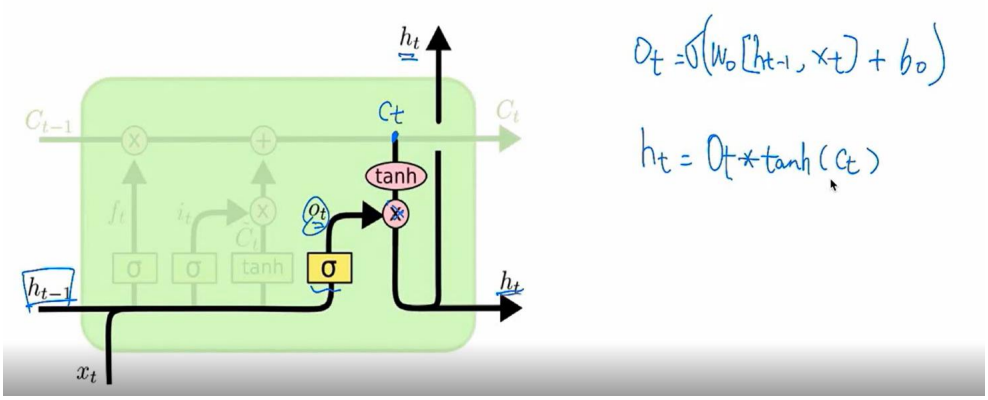
如图所示，it 也是用 sigmoid，形成一个门机制，与 c(t)_相乘过滤，得到下一时刻的过滤后的预输出信息；让我们再看看 c(t)_的结构，有没有眼熟，这不就是 rnn 的结构？生成的 c(t)_就是下一时刻的预输出。



c 的更新图

结合遗忘门与输入门，遗忘门得到过滤后 c(t-1)信息+过滤之后的当前预输出信息 c(t)_,完成 c 的更新。

③输出门



输出门则是对更新好的 C(t)在进行一次过滤，生成真正的输出。

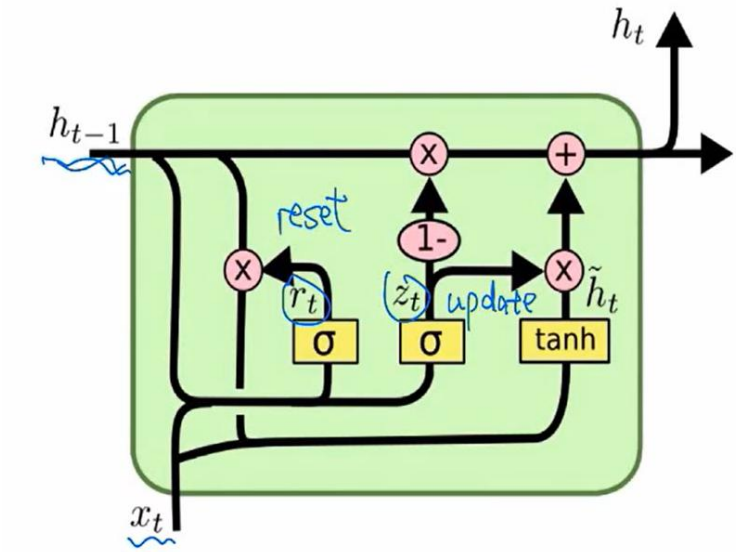
④为何能解决梯度消失？

答：主要体现 c(t)的更新用了一个”+”号，在反向更新时，会减缓一定的梯度消失。

减缓梯度消失的方式：更换激活函数（relu）、bn、使用残差结构

减缓梯度爆炸的方式：梯度截断

3、GRU 结构



gru 结构图

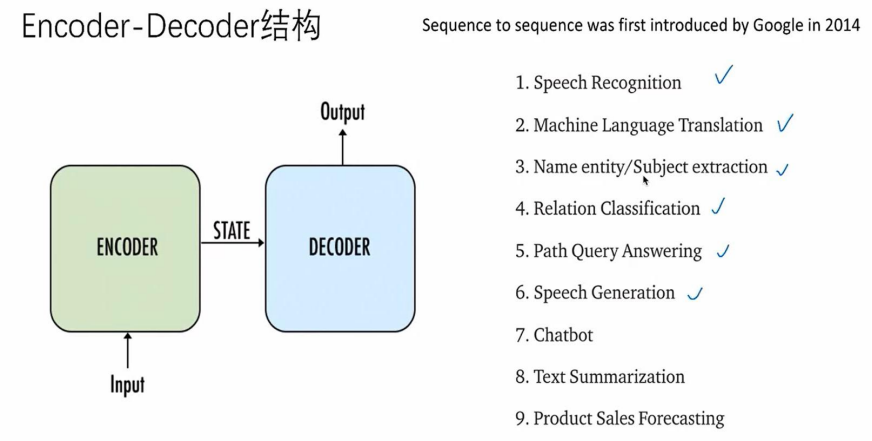
gru 有两个门：重置门和更新门。重置门负责对 h(t-1)的过滤，看到“乘”节点，表示将(1-zt)对 h(t-1)过滤，zt 对预输出 h(t)_进行过滤，然后相加，得到 h(t)；仔细一看，与 lstm 如出一辙。h(t)_是由过滤后的 h(t-1)与 x_t 融合后得到，这就是 rnn 的结构，与 lstm 基本一致。

lstm 与 gru 的比较？

答：从结构上来看，gru 比 lstm 少用一个门，参数更少，速度更快；lstm 则因为参数更多，效果更好。实际运用，gru 用的比较多，效果上基本没差别。

4、Seq2Seq 结构

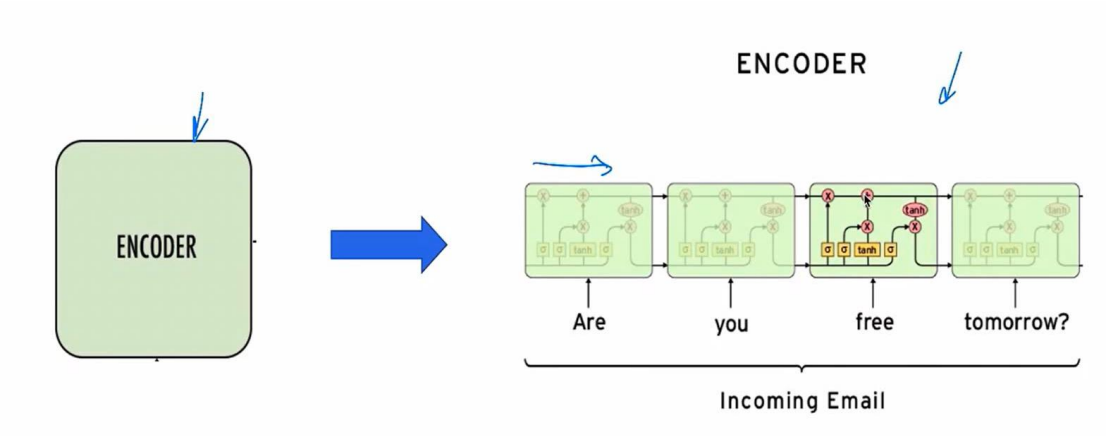
Seq2seq 是一个 encoder-decoer 结构，encoder 用于做编码，decoder 用于做解码生成，整体结构如下图：



Seq2seq 结构图

4.1 encoder 结构

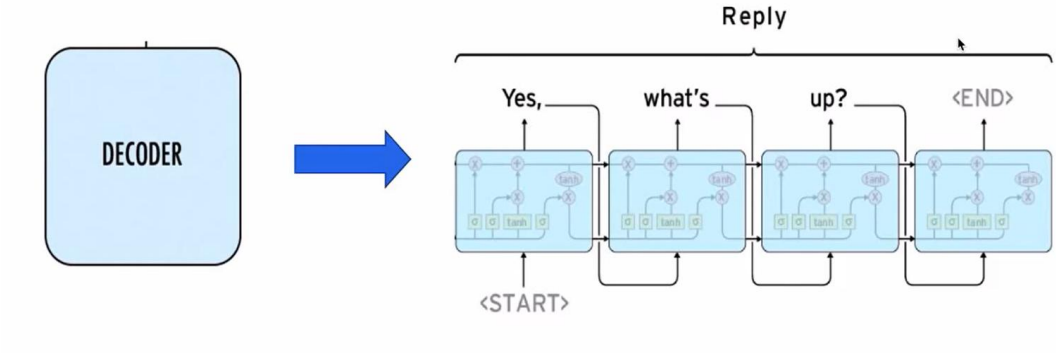
Encoder 若用于对序列的编码，一般采用 rnn(可双向)等结构，具体结构视业务而定，以下是 encoder 结构图，仅供参考：



Encoder 结构图

4.2 decoder 结构

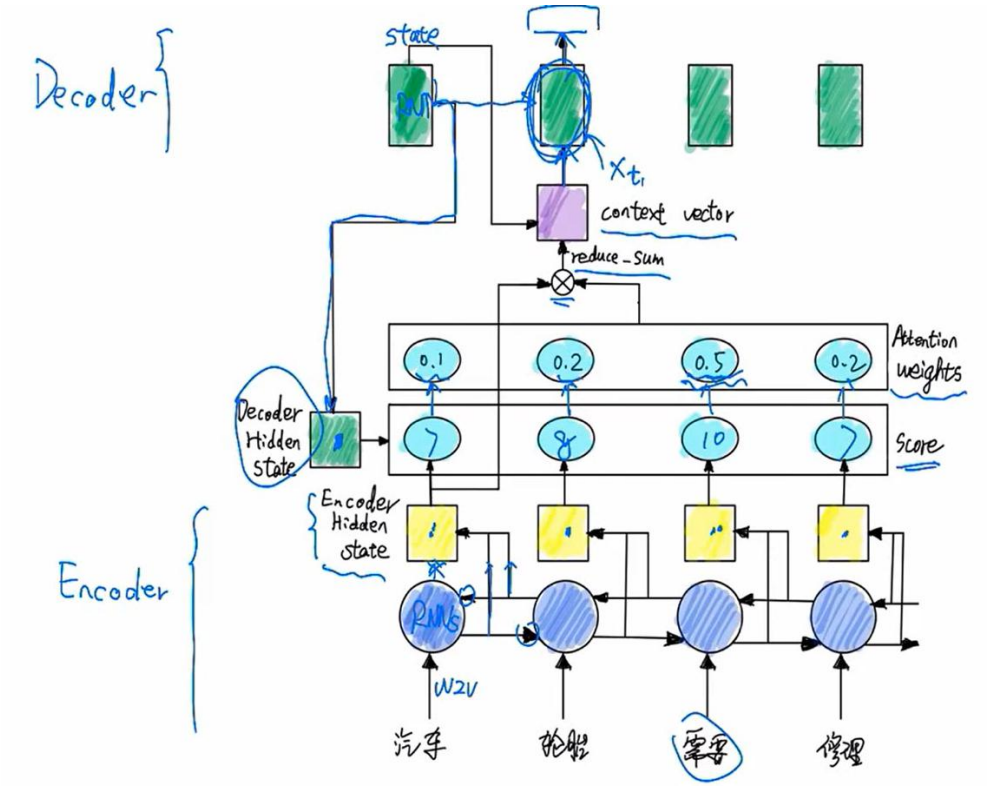
decoder 若用于对语义的解码与生成，结构一般采用单向 rnn，在输入上与 encoder 不同，通常采用<START>作为首次输入，<END>作为截止。



二、Attention 原理与应用

encoder 编码后的语义一般最后一个 cell 的 c(t)，这就会出现一个问题，当序列很长的时候，整个句子的语义就会很弱。Attention 的提出，给定输入序列一定的权重，有侧重

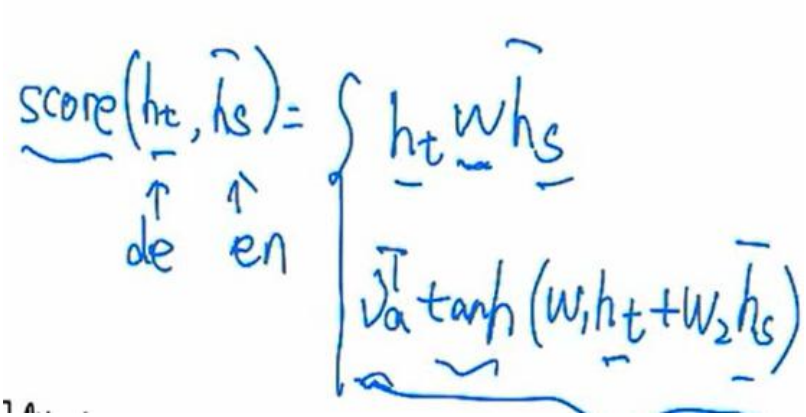
的关注影响力大的部分，对效果提升有很大帮助。



根据上图，对 Attention 进行分析，①encode 部分会生成每一时刻的 hidden_state ②decode 的上一个输入 o(t-1)（0 时刻选用 encode 语义）和 hidden_states(所有 hidden_state)进行融合，得到 score; ③将 score 进行归一化，得到 attention_weight; ④将所有 hidden_state 分别乘上 attention_weight，之后 reduce_sum，得到 context_vector; ⑥将上一时刻的 o(t-1)与 contenxt_vector 融合作为下一时刻的输入。

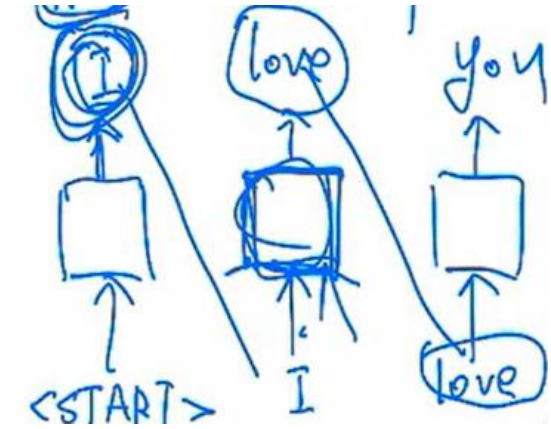
注：reduce_sum 操作：假设 bs=1,hidden_states=[enc_len,hidden_size],因为有 enc_len 个,attention_weiht=[enc_len],经过 reduce_sum 之后，enc_len 会被消掉，结果为[1,hidden_size].

Score 计算方式？



Score 的目的是求相似度，具体方式：①直接点乘 h_t 与 h_s，这种方式没有训练参数，不太好；②引入 w 再点乘，如上图，可以训练 w;③MLP 融合方式，本次项目用这一种。

Attention 为啥用上一时刻输出去求 score?



从上图可看出，这是 decoder 的结构，<START>生成“I”，而“I”又得做下一次的输入，因此得出，即使生成的不是“I”，也是与“I”非常相似的单词，所以，用上一时刻的输出去求 score.

三、Layer 与 Model 模型搭建

Layer、model 能很好的区分与管理模型，seq2seq 这个项目可以把 encoder、decoder 用 layer 来做，seq2seq 整体结构用 model 来做。

四、训练过程中的 Exposure Bias

- 训练的方式：Teacher forcing

  答:一般情况下,seq2seq 的 decoder 生成,会采用<START>生成一个 word,再将当前生成 word 作为下一次输入;Teacher forcing 则是将 target_label 直接作为下一次输入。

- Teacher forcing 的好处与坏处

  答：好处：直接采用 target_label 作为下一次输入，避免模型一开始就跑偏，Teacher forcing 能解决收敛速度慢和不稳定问题

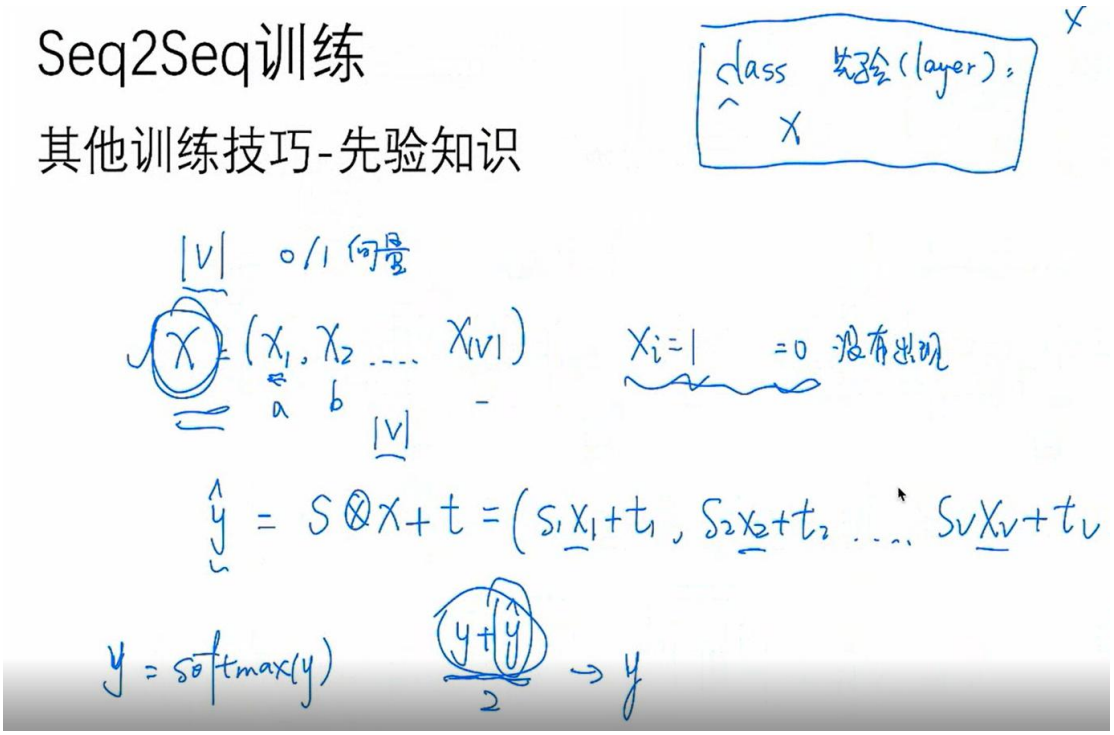
  坏处：容易产生 Exposure Bias。Exposure Bias（曝光偏差）指在训练时拟合很好，预测时就表现极差，又好比：一个学生在老师一点一点，考试时，不按套路来，扔一个老师没教过的题，就做不出来（泛化能力差）。

- 解决曝光偏差方式

  数学角度解释:生成问题是一个自回归问题,当前 word 与前 n-1 个 word 都有一定的连续性，在求概率是采用联合概率，然而每一个预测的概率都会或多或少的偏差，通过连乘之后，偏差越来越大。

  解决策略：①对抗策略（加干扰）②预测时用 beam_search ③训练时根据 epoch 做一个衰减系数 x，decode 输入=（1-x）*生成 label+x*target_label.

- 训练拔高点



我们在做摘要时，往往会先拿原文中的词来做摘要结果，因此，我们可以判断 word 是否在原文出现作为先验知识。

具体做法：假设 encode 输入为 context，摘要为 label，vocab 词典 V 是 context 与 label 总和的值；如上图，步骤解析：①手工总结一个向量 X，size=vocab_size，这个向量每一次输入统计一次，根据 vocab 词典 V 中的每一个词，如果在当前 context 中出现过，设为 1，反之设置 0，由此得到一个[vocab_size]的向量（batch_size=1） ②定义一个参数 S 与 t，进行矩阵乘，训练参数，并且得到先验 y_； ③再将(y_+y)/2 得到预测结果。