Transformer相关——（9）训练Transformer

发表于 2021-08-18 | 更新于 2021-08-18 | 深度学习 | 字数总计: 1,556 | 阅读时长: 6分钟 | 阅读量: 5073

Transformer相关——（9）训练Transformer

引言

现在已经对Transformer的前向传播过程了解比较清晰了，这一篇总结一下Transformer模型的训练和预测过程。主要参考了李宏毅老师的21年春季的课程。

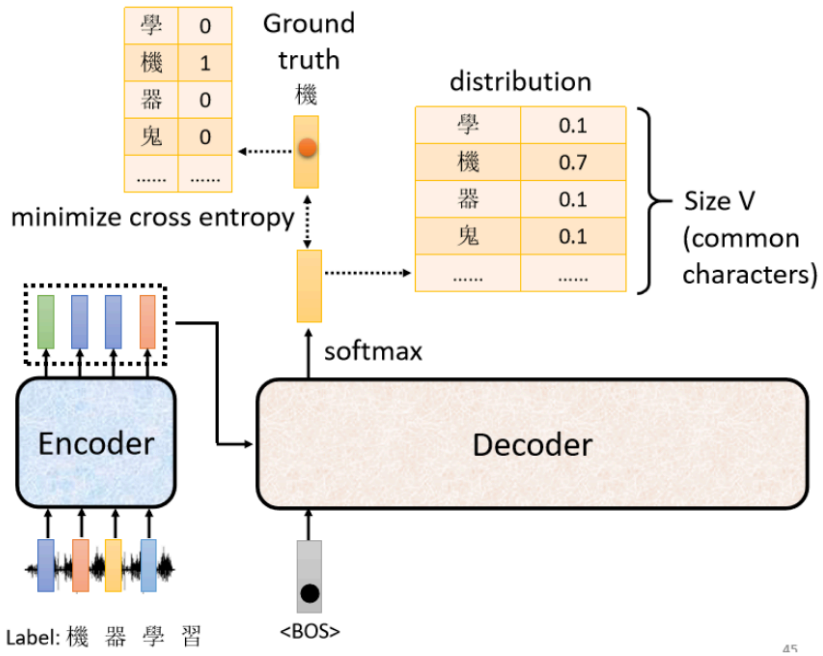
Transformer的Loss function

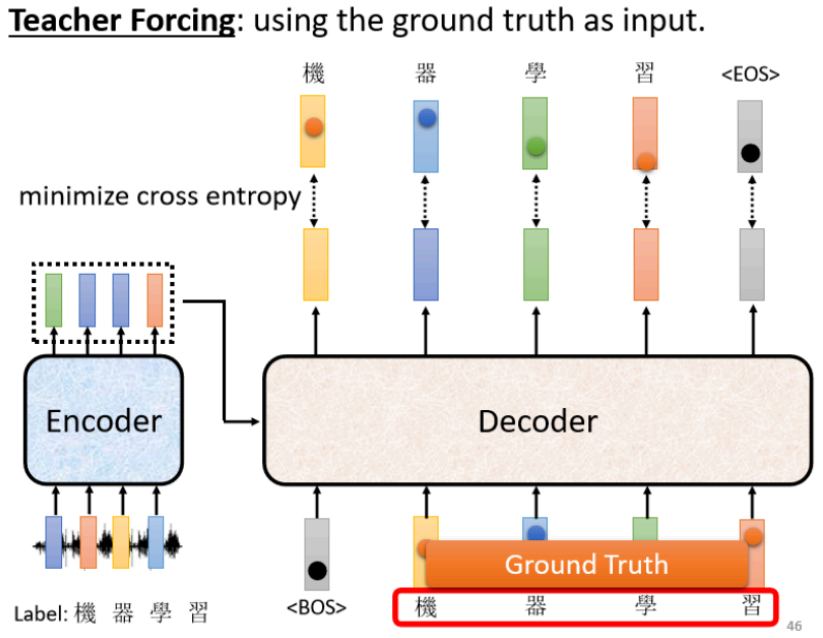
以语音识别任务为例，每一个语音识别过程实际上和分类任务很像。

1. Decoder的输出经过一个输出维度大小等于字典（或者说类别）的线性层，再经过一个softmax层求得**各个词（或者说类别）的概率**

分布；

2 然后计算每一个词的概率分布和 Ground Truth之间的 Cross Entropy（Cross Entropy是分类常用的损失函数，其他任务损失函数的选择可参考：深度学习中常见的激活函数与损失函数的选择与介绍），每一个位置的预测都相当于是一次分类，最终计算一个batch总和的Cross entropy，minimize这个 Cross Entropy 的值。



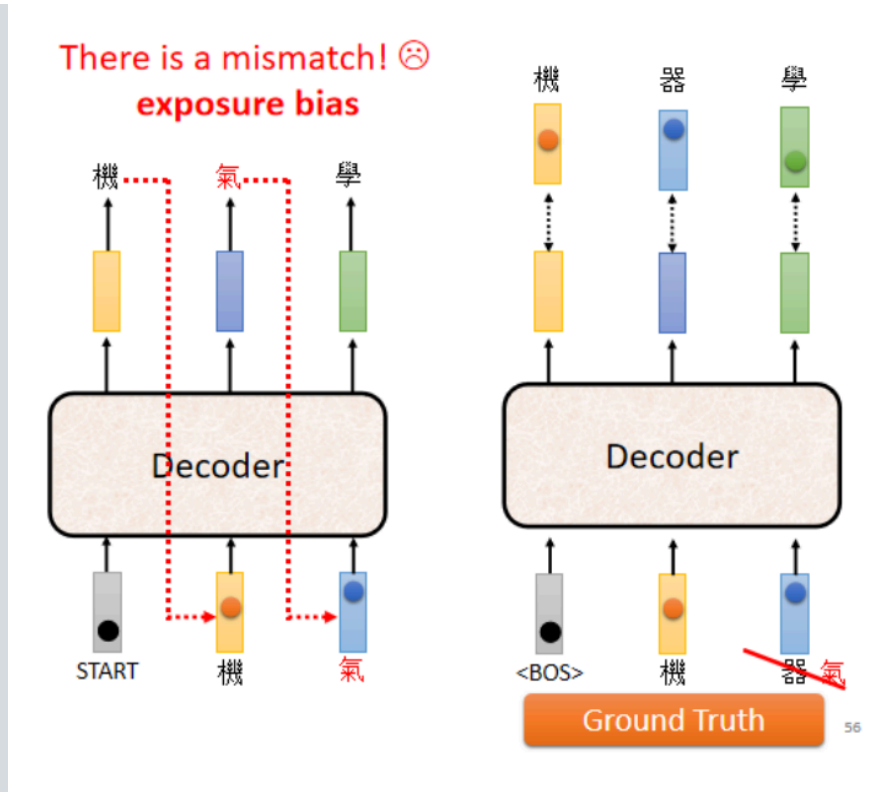


由上图可以看到，在训练的时候，**Decoder在输入的时候就给了正确答案（直接逐步喂入目标序列的embedding，这种训练方法也叫作 Teacher Forcing）**。但是**在预测的时候，是没有正确答案的，看到的是自己的前一个输出**。如果前一个输出错了，很可能会导致后面的输出也接连错误（**误差累积**，一步错，步步错）。

解决该问题的一个策略是**scheduled sampling计划采样**。

scheduled sampling计划采样

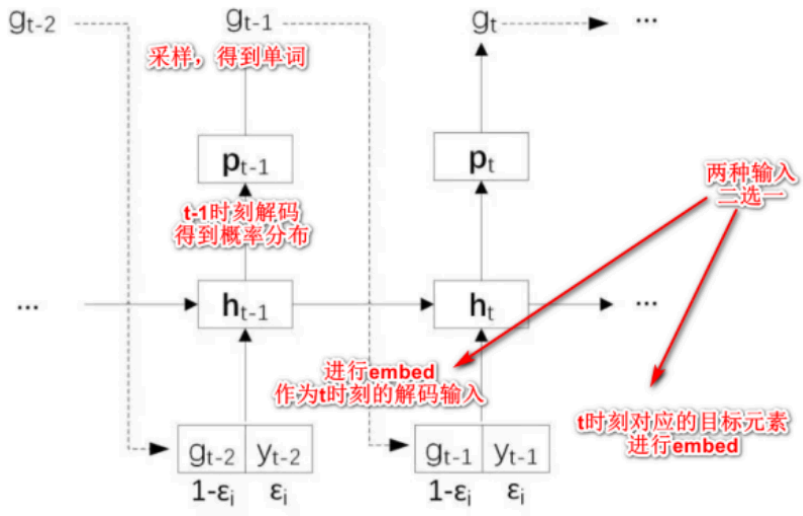
scheduled sampling策略主要应用在序列到序列模型的训练阶段，而生成阶段则不需要使用。其基本思想，在训练的时候，我们就给Decoder的输入加入一些错误的东西，让它正确预测结果。



实现原理

设置一个概率值，决定当前解码的输入来自于以下二选一：

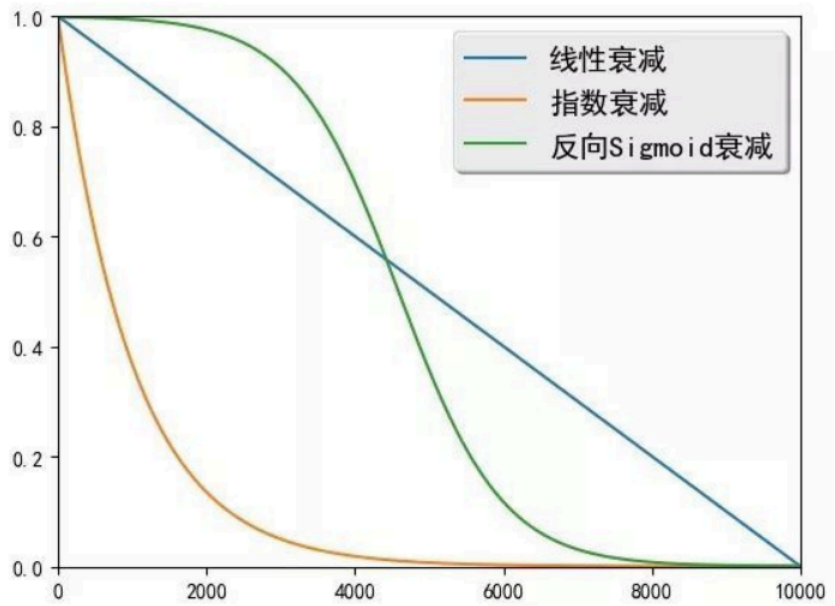
1. 真实的目标序列元素（ground truth）；
2. 上一时刻模型的预测结果（可能是对的也可能是错的）。



这个概率值可以使用衰减函数随着训练次数进行变化，假设有ϵ_i的概率使用上一时刻的真实元素作为解码器输入，那么常见的衰减方式有：

- 线性衰减：$\epsilon_i = max(\epsilon, k - c * i)$，其中$\epsilon$限制$\epsilon_i$的最小值，k和c控制线性衰减的幅度。
- 指数衰减：$\epsilon_i = k_i$，其中$0 < k < 1$，k控制着指数衰减的幅度。
- 反向Sigmoid衰减：$\epsilon_i = k/(k + exp(i/k))$，其中$k > 1$，$k$同样控制衰减的幅度。

  解码器将不断倾向于使用生成的元素作为输入，训练阶段和生成阶段的数据分布将变得越来越一致。



缺点

会影响到Transformer的并行化能力。

将scheduled sampling应用到Transformer上的训练技巧可参考：

Scheduled Sampling for Transformers

Parallel Scheduled Sampling

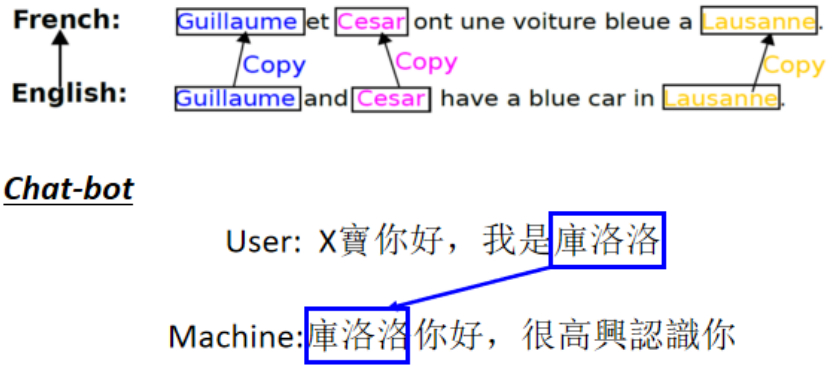
训练TIPS（Seq2Seq模型都适用）

根据不同的下游任务有一些特别的训练技巧。

copy mechanism

普通的Transformer要求 Decoder 自己产生输出，但是对很多任务而言，也许 **Decoder 没有必要自己产生输出**，而是可以**从输入的序列中复制**一些东西出来。

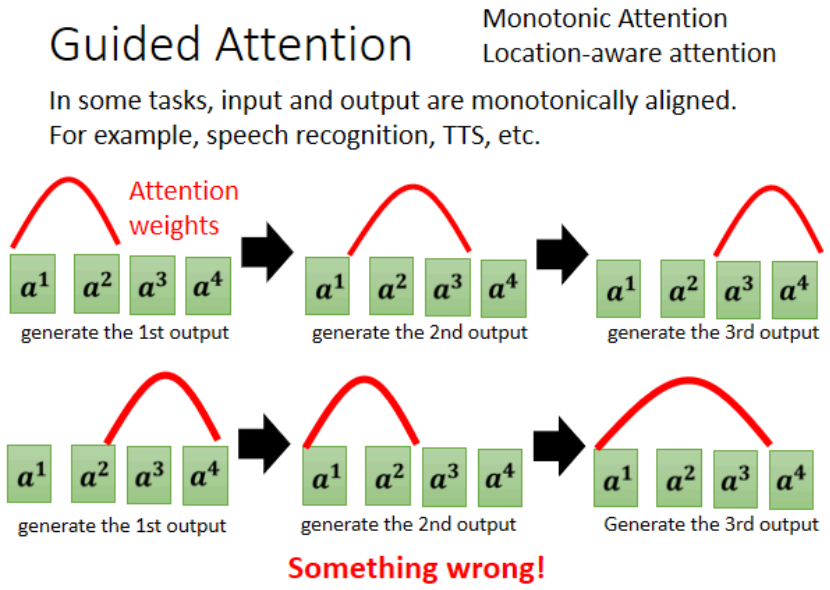
这个**策略可以用于训练聊天机器人、文章摘要提取等任务**。比如说一个非常罕见的词汇在训练数据中可能一次也没有出现过，那Decoder不太可能能正确地生成这段词汇。



guided attention

这个训练策略在于，有一些经验知道attention的分布大致长什么样，于是要求**机器它在做 Attention 的时候,是有固定的方式的**。

比如在语音识别中，从左到右说每个字，输出的每个字对语音的attention分布其峰值就应该是从左往右移动，而不是在左右反复横跳。

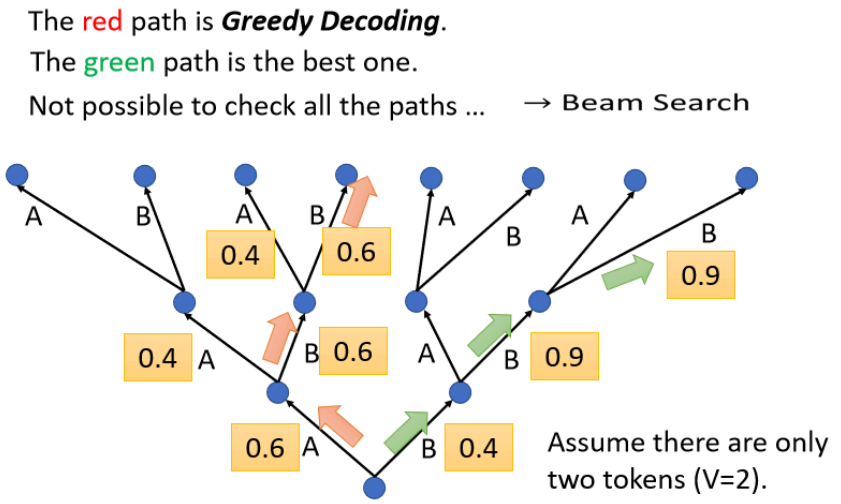


Beam Search

在之前文本翻译任务中，**每次都选择概率分布最大的词输出，这是greedy decoding贪婪解码**。

而beam search是找一个 Approximate，选一个估测的结果，找一个不是很精确的Solution。

beam search在**每个时间步保留n个最高概率的输出词，然后在下一个时间步，重复执行这个过程**：假设beam_size为2，第一个位置概率最高的两个输出的词是”I“和”a“，这两个词都保留，然后根据第一个词计算第二个位置的词的概率分布，再取出 2 个概率最高的词，对于第二个位置和第三个位置，重复这个过程。



Beam Search对什么任务有效呢?

看任务的本身的特性

- 假设一个任务的**答案非常地明确，通常 Beam Search 就会比较有帮助**

  举例来说明答案非常明确是什么意思。比如说语音识别，说一句话识别的结果就只有一个可能，就那一串文字就是你唯一可能的正确答案，并没有什么模糊的结果。

- **需要机器发挥一点创造力的时候， Beam Search 可能会失灵**

  举例来说比如Sentence Completion任务，给一个句子或者一个故事的前半段，后半部有无穷多可能的发展方式，那这种需要有一些创造力的，有不是只有一个答案的任务，往往会比较需要在 Decoder 里面，加入随机性；语音合成任务TTS ，也需要加入一些随机性。

用强化学习训练

loss function和最终的评价指标往往并不是相同的，但是评价指标的计算方式可能很复杂是不可微分的（没有办法梯度求导反向传播），比如BLEU Score（bilingual evaluation understudy，双语互译质量评估辅助工具）。

那么可以用RL（reinforce learning），**将评价指标作为RL的reward，把decoder当作agent进行训练**。

参考文献

(强推)李宏毅2021春机器学习课程

李宏毅老师机器学习课程笔记

2.2-图解transformer.md

【序列到序列学习】使用Scheduled Sampling改善翻译质量

李宏毅自然语言处理——Transformer