在使用tensorflow时发现其提供了两种Attention Mechanisms（注意力机制），如下

The two basic attention mechanisms are:

tf.contrib.seq2seq.BahdanauAttention (additive attention, ref.)

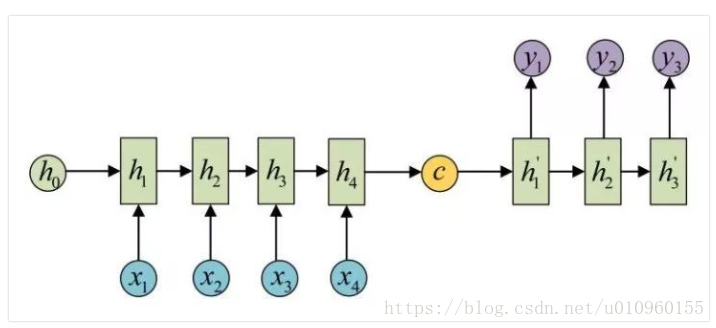
tf.contrib.seq2seq.LuongAttention (multiplicative attention, ref.)

那么这两种注意力机制有何异同呢？下面我们通过两个方面介绍，首先简单回顾下注意力机制，然后再对这两种经典的注意力机制进行比较。

一、注意力机制回顾

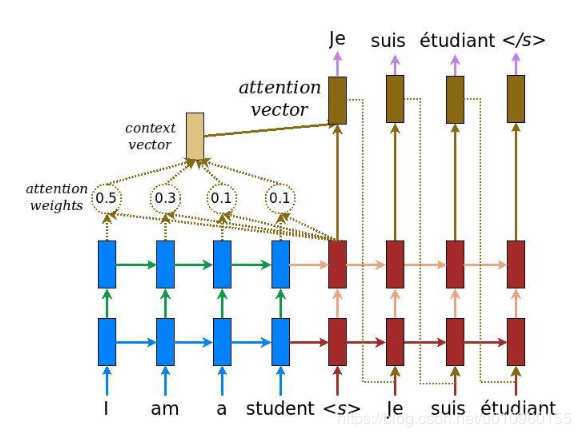
简单来说，注意力本质上就是一个经过softmax层输出的向量。

在早期机器翻译应用中，神经网络结构一般如下图，是一个RNN的Encoder-Decoder模型。左边是Encoder，代表输入的sentence。右边代表Decoder，是根据输入sentence对应的翻译。Encoder会通过RNN将最后一个step的隐藏状态向量c作为输出，Deocder利用向量c进行翻译。这样做有一个缺点，翻译时过分依赖于这个将整个sentence压缩成固定输入的向量。输入的sentence有可能包含上百个单词，这么做不可避免会造成信息的丢失，翻译结果也无法准确了。



注意力机制的引入就是为了解决此问题，注意力机制使得机器翻译中利用原始的sentence信息，减少信息损失。在解码层，生成每个时刻的y，都会利用到x1,x2,x3....，而不再仅仅利用最后时刻的隐藏状态向量。同时注意力机制还能使翻译器zoom in or out（使用局部或全局信息）。

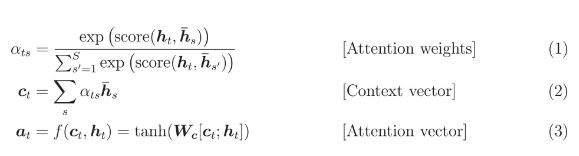
注意力机制听起来很高大上、很神秘，其实它的整个实现只需要一些参数和简单的数学运算。那么注意力机制到底是如何实现的呢？



在基本的Encoder-Decoder模型中，注意力机制在Encoder和Decoder加入了上下文向量context vector，如上图所示，左边蓝色的代表Encoder，红色的代表Decoder。对于Decoder中每个要生成的y，都会生成一个上下文向量。这个上下文向量是由每个输入的words的信息加权求和得到的，其中权重向量就是注意力向量，它代表在此刻生成y时输入的单词的重要程度。最后将上下文向量和此刻的y的信息进行融合作为输出。

构建上下文向量过程也很简单，首先对于一个固定的target word，我们把这个target state跟所有的Encoder的state进行比较，这样对每个state得到了一个score;然后使用softmax对这些score进行归一化，这样就得到了基于target state的条件概率分布。最后，对source的state进行加权求和，得到上下文向量，将上下文向量与target state融合作为最终的输出。

具体流程的数学表达如下：



为了理解这个看起来有些复杂的数学公式，我们需要记住三点：

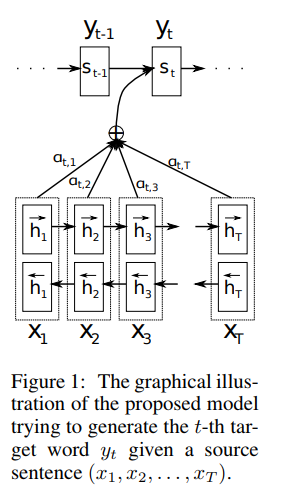
1. 在解码时，对于每个输出的word都需要计算上下文向量。所以，我们会得到一个的2D矩阵， 代表source word数量，代表target word数量
2. 我们可以通过context vector,target word,attention function 计算attention vevtor

3. attention mechanism是可以训练的。

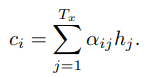
二、BahdanauAttention与LuongAttention

2.1 BahdanauAttention

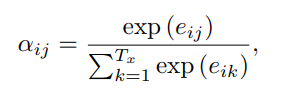
BahdanauAttention是Bahdanau在论文NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE中提出的，整体Attention结构如下图：



1）第个target word上下文向量会根据每个source word的隐向量加权求和得到：



2）对于每个 的计算如下



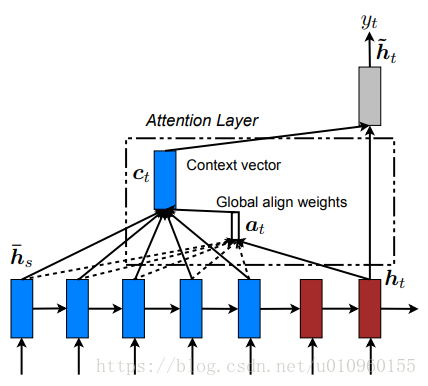
其中

https://img-blog.csdn.net/20180926172008662?watermark/2/text/aHR0cHM6Ly9ibG9nLmNzZG4ubmV0L3UwMTA5NjAxNTU=/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70

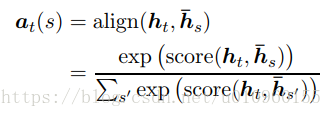
* 是对齐模型，代表位置的输入和位置的输出匹配程度的分数，这个分数基于RNN的 i-1 位置的隐含状态https://private.codecogs.com/gif.latex?s_%7Bi-1%7D和 j 位置的https://private.codecogs.com/gif.latex?h_%7Bj%7D计算得到。

2.2 LuongAttention

LuongAttention是Luong在论文Effective Approaches to Attention-based Neural Machine Translation中提出的。整体结构如下

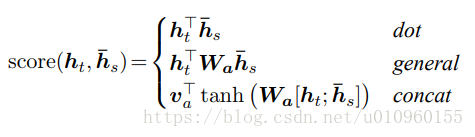


 与BahdanauAttention整体结构类似，LuongAttention对原结构进行了一些调整，其中Attention向量计算方法如下



 其中与BahdanauAttention机制有以下几点改进：

1. BahdanauAttention对Encoder和Decoder的双向的RNN的state拼接起来作为输出，LuongAttention仅使用最上层的RNN输出
2. BahdanauAttention的计算流程为 ht−1 → at → ct → ht，它使用前一个位置t-1的state计算t时刻的ht。LuongAttention计算流程为  ht → at → ct → h˜t 使用t位置的state当前位置的ht
3. BahdanauAttention只在concat对齐函数上进行了实验，LuongAttention在多种对齐函数进行了实验，下图为LuongAttention设计的三种对齐函数



2.3 总结

BahdanauAttention与LuongAttention两种注意力机制大体结构类似，都是基于第一节中的attention框架设计，主要的不同点就是在对齐函数上，在计算第 个位置的score，前者是需要使用https://private.codecogs.com/gif.latex?s_%7Bi-1%7D 和https://private.codecogs.com/gif.latex?h_%7Bj%7D 来进行计算，后者使用https://private.codecogs.com/gif.latex?s_%7Bi%7D和https://private.codecogs.com/gif.latex?h_%7Bj%7D计算，这么来看还是后者直观上更合理些，逻辑上也更顺滑。两种机制在不同任务上的性能貌似差距也不是很大，具体的细节还待进一步做实验比较。