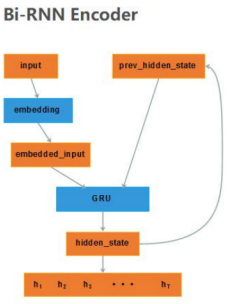
2.Attention

、　由于Seq2Seq的结构中，编码器将序列信息提取成为一个向量作为解码器的解码信息，这样存在2个问题：1.当解码器在生成输出序列时可能只需利用输入序列某一部分的信息，而编码器只提供一个最终向量输出，这样很难提取；2.对于长语句编码，由于定长向量的转化会产生信息损失。

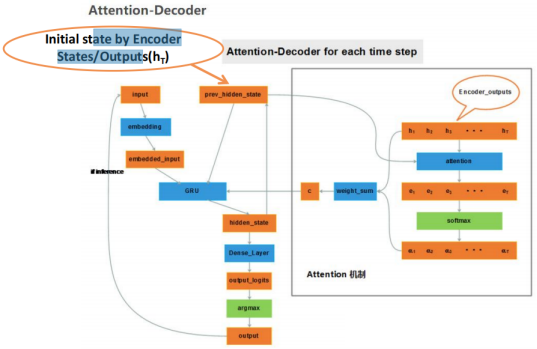
Attention的出现，就能在解决Seq2Seq中存在的定长序列导致的信息损失的同时，将注意力集中在部分所需关注的输入序列。

1.Attention的第一种理解方式

编码器和Seq2Seq一样，不做改变，如下图所示（**编码器可以是双向的，但是解码器不能是双向的**）。



在**解码器**中，基于每个时刻编码器的输出和上一个时刻解码器的输出来构建AttentionScores：$e_{t,i} = F(h_i, s_{t-1}), e = (e_{t,1}, e_{t,2}, ..., e_{t,n})$,然后基于AttentionScores得到关注度的概率分布$\alpha_t = softmax(e_t)$，最后基于概率分布和编码器的状态信息计算出Attention值：$Attention_t = \sum_{i=1}^{n} \alpha_{t,i} h_i$，再将Attention值和解码器当前时刻的输入值拼接$[y_t; Attention_t]$。



F（.)在不同论文有不同的表示：

乘法Attention：

$$e_{t,i} = s_{t-1}^T h_i$$

$$e_{t,i} = s_{t-1}^T h_i / \sqrt{d}$$

加法Attention：

$$e_{t,i} = s_{t-1}^T W h_i$$

$$e_{t,i} = u^T \tanh(W_1 h_i + W_2 s_{t-1})$$

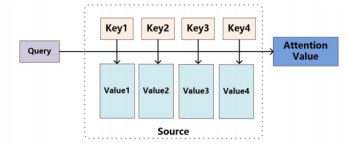
$$e_{t,i} = W_1 h_i + W_2 s_{t-1}$$

$$e_{t,i} = W h_i$$

TensorFlow默认

2.Attention的第二种理解方式（QKV）

将解码器上一个时刻的输出作为查询条件Query，编码器各个时刻的输出作为关键词信息key和在查询中需要获得的编码器信息Value，通过计算Query和各个Key的相似性或者相关性，得到每个Key对应Value的权重系数，然后对Value进行加权求和，即得到了最终的Attention数值。所以本质上Attention机制是对**Source中元素的Value值进行加权求和，而Query和Key用来计算对应Value的权重系数。**



$$\text{Attention}(\text{Query}, \text{Source}) = \sum_{i=1}^{L_x} Similarity(Query, Key_i) * Value_i$$

相似度的计算方法：

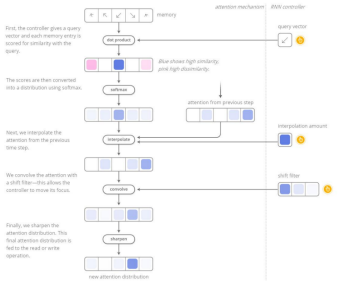
点积： $\text{Similarity}(\text{Query}, Key_i) = \text{Query} \cdot Key_i$

Cosine 相似性： $\text{Similarity}(\text{Query}, Key_i) = \frac{\text{Query} \cdot Key_i}{||Query|| \cdot ||Key_i||}$

MLP 网络： $\text{Similarity}(\text{Query}, Key_i) = \text{MLP}(\text{Query}, Key_i)$

3.Attention的第三种理解方式

基于向量方向的余弦相似度。



4.Transformer中的Attention

不同于Seq2Seq仅把attention用于Decoder，Transformer将attention同时用于Decoder和Encoder中，下面是Transformer中的Attention细节：

1. Self Attenion:Seq2Seq中使用attention是为了利用解码器当前时刻的信息和编码器所有时刻的信息对编码器所有时刻的信息进行加权求和，从而获得解码器当前时刻的输入信息。而transformer中的self-attention是为了提取编码器每个时刻的单词在上下文中的信息。
2. MultiHead Attention：transformer使用了多头注意力机制，在保证不改变参数量的情况下，提取更多方面的上下文特征，增强了attention的表达能力。
3. Masked Attention:Transformer在Decoder中利用了Masked Attention，这是为了防止decoder在解码encoder层输出时“作弊”，提前看到了剩下的答案，所以强迫模型根据输入序列之前的结果进行attention。
4. Cross Attention:Transformer中在对编码器的解码过程中，使用了CrossAttention，但是在transformer中的CrossAttention和Seq2Seq中却有所不同，Seq2Seq进行解码时，使用的是当前解码时刻上一时刻的隐藏层输出信息作为q，而transformer则是使用当前解码时刻的输入信息作为q；Seq2Seq当前解码时刻的输入信息是编码器各个时刻的加权和与当前时刻的输入信息的拼接，Transformer当前解码时刻的输入信息是编码器各个时刻的权重和。