Attention机制

什么是attention机制？

为什么需要attention机制？

attention机制的实现方法？

Grissom, 2019/07/07

什么是attention机制？

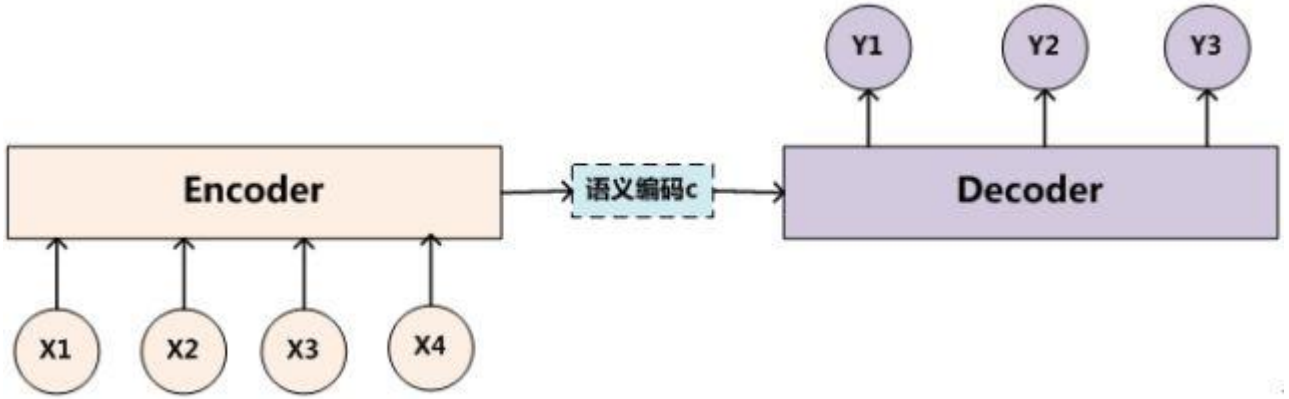
“他是一位三十多岁的程序员， 没日没夜的加班使他严重脱发。”
“He is a software engineer in his 30s that ...”

我们在翻译“程序员”的时候，只需要关注这个词及其**附近的词**或**相关的词**即可，并不需要记住全部句子再去翻译。这就是所谓的“attention机制”。

为什么需要attention机制？（1）

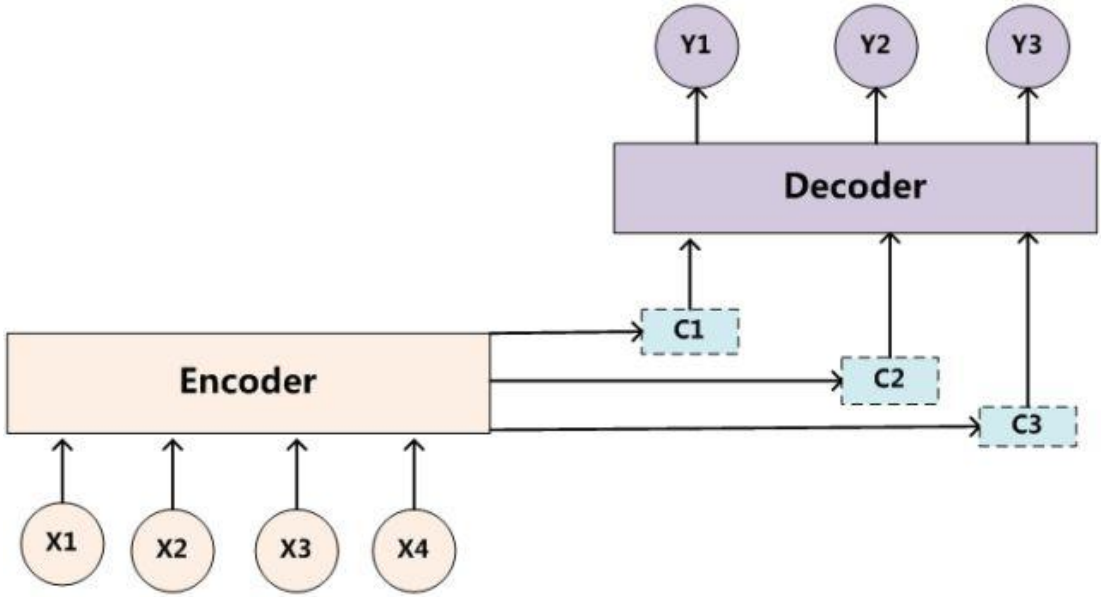
为了解决我们在seq2seq中遇到的问题。

以机器翻译为例：在seq2seq中，我们是这样处理的（如下图）



存在的问题：
1）因为语义编码c是定长的，当输入到Encoder中的句子（X序列）很长时，必然会造**部分信息丢失**。
2）当Decoder生成句子（Y序列）时，每个词（Y_i）用到的语义编码c是完全相同的，**难以区分当前要翻译的是哪一个输入词**。

为什么需要attention机制？（2）



假设现在为每个词生成对应的语义编码C_i，使翻译行为更有针对性（知道翻译的是原句中的哪个词），翻译的效果自然会更好。 [即：用C_i来表征不同的attention向量]

* 注意：原句单词与译文单词的位置并不需要严格的一一对应，可以有些许不一致。但大体还需在其附近的，因为RNN只能保留附近的词义，若离得太远旧词词义可能会被覆盖掉。

attention机制的实现方法（1）

那么，如何为每个词生成对应的“语义编码c_i”呢？

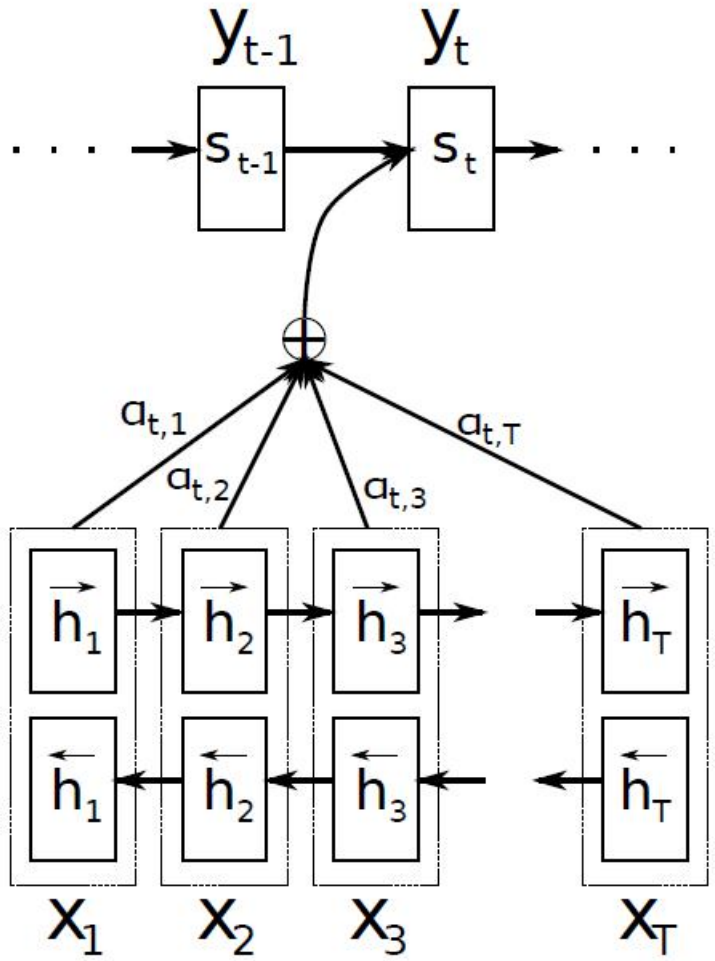
Step01 - 通过一个双向RNN来学习到**每个词前向、后向词信息**

$\overrightarrow{h}_j^\top$ forward-RNN在第j个单词上的含义向量（包含了1~j个单词的某种含义）

$\overleftarrow{h}_j^\top$ backward-RNN在第j个单词上的含义向量（包含了j~T个单词的某种含义）

然后将两者拼接，**得到单词j的词向量**

$$h_j = \left[\overrightarrow{h}_j^\top ; \overleftarrow{h}_j^\top\right]^\top$$



** 注意：上标T表示向量转置（transposition），下标T表示输入的句子中有T个单词。*

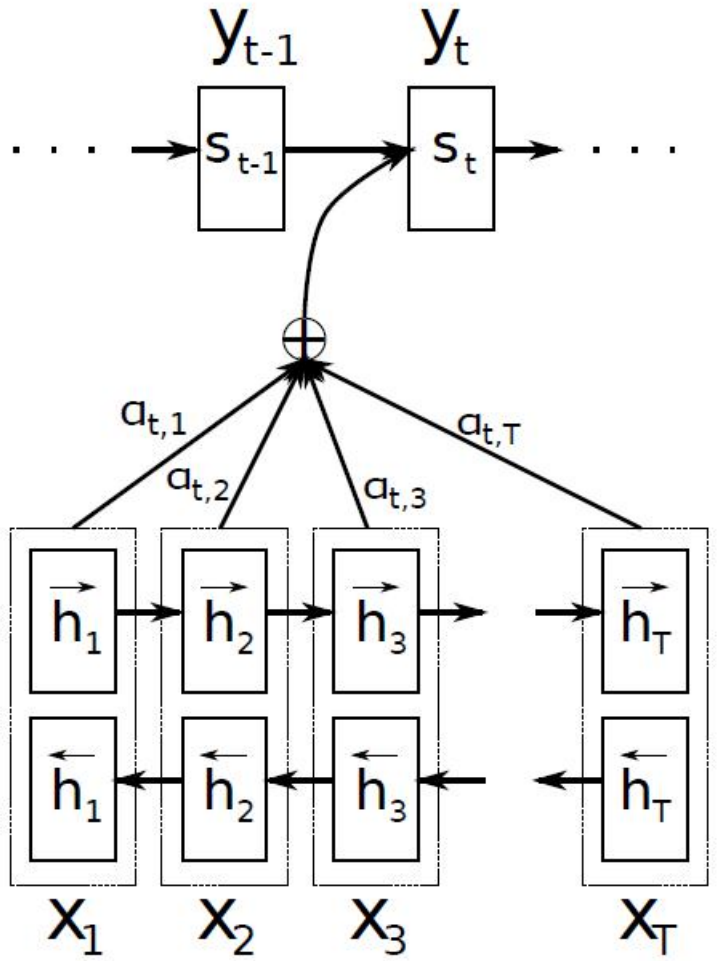
attention机制的实现方法（2）

Step02 - 使用函数a来度量**原句中单词j的词向量**与**译文中第i-1时刻隐状态向量**的相似度。

$$e_{ij} = a(s_{i-1}, h_j)$$

s_{i-1} 包含着译文中已翻译出来的前i-1个词的含义

h_j 包含着原句前1~j个单词的含义，及j~T个单词的含义



*** 这么说的话，是不是仅用forward-Rnn就够了呢？**

*** 另外，这里的向量S的维度岂不是要设定为h向量的2倍长？**

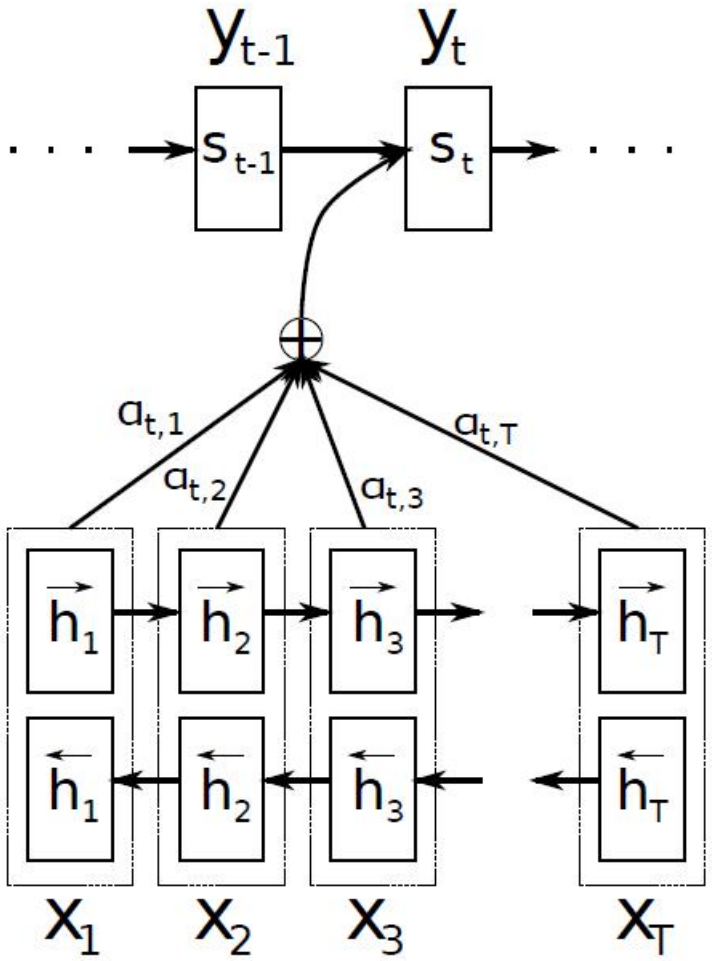
attention机制的实现方法（3）

Step03 - 计算其与原句中每一个单词的向量相似度，并归一化

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

这里的i，是指译文中第i个单词。
[它的隐状态向量包含了已翻译出来的前1~i个单词的词义]

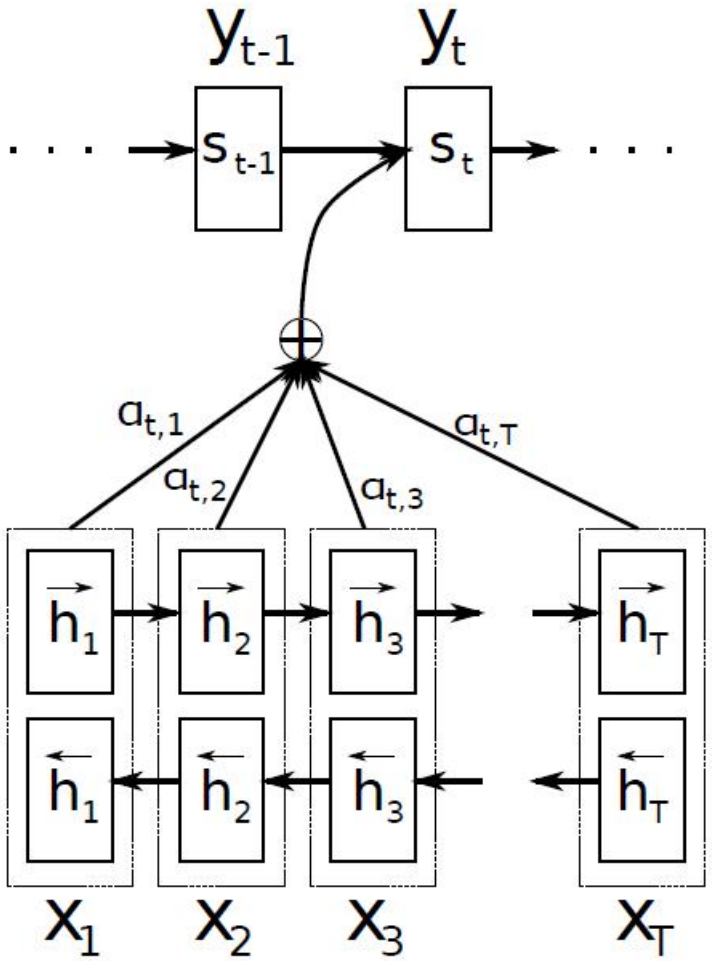
这里的j，是指原句中第j个单词。
[它的词义向量由两部分组成：前1~j个词的词义; 后j~T个词的词义]



attention机制的实现方法（4）

Step04 - **原句中每个单词的向量 * 其对应的相似度**，得到最终值
（我们可以认为这是在“求期望”）

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$



那么，这个函数a是什么呢？

有论文上说它是alignment model（**feedforward neural network**），也有论文说它是a learnable function，但本质上它是一个自定义的“计算两个向量相似度”的函数。一般使用如下三种：

点积： $\mathbf{Similarity}(\mathbf{Query}, \boldsymbol{Key}_i) = \mathbf{Query} \cdot \boldsymbol{Key}_i$

Cosine 相似性： $\mathbf{Similarity}(\mathbf{Query}, \boldsymbol{Key}_i) = \frac{\mathbf{Query} \cdot \boldsymbol{Key}_i}{\|\boldsymbol{Query}\| \cdot \|\boldsymbol{Key}_i\|}$

MLP 网络： $\mathbf{Similarity}(\mathbf{Query}, \boldsymbol{Key}_i) = \mathbf{MLP}(\mathbf{Query}, \boldsymbol{Key}_i)$

* 这里的 $\mathbf{Query}$ 是上文中的 s_{i-1}
* 这里的 $\boldsymbol{Key}_i$ 是上文中的 h_j

需要注意的是，在论文《NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE》中，作者说的是the alignment model **a** as **a feedforward neural network** which is jointly trained with all the other components of the proposed system...the alignment model directly computes a soft alignment, which allows the gradient of the cost function to be backpropagated through. This gradient can be used to **train the alignment model** as well as the whole translation model **jointly**.