Seq2Seq + Attention

1. Seq-to-Seq框架1

第一个要介绍的Seq-to-Seq模型来自于“Learning Phrase Representations using RNN Encoder-Decoder for Statistical Maching Translation”,其结构如下图：

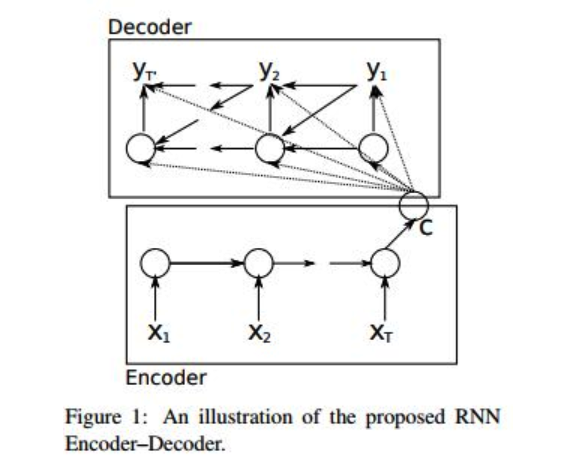


图1： Seq-to-Seq结构

模型在encoder过程中将整个source序列编码成一个固定维度的向量c(也就是RNN最终的隐藏状态), 其中

其中的f就是各种RNN模型。并且认为c中保存了source序列的信息，将其传入decoder阶段即可，**在每次进行decoder时，仍然使用RNN神经元进行处理（下面公式中的f函数）。此时的输入包括前一时刻的输出，前一隐层状态的,输入编码向量c三个向量，也就是解码时同时会考虑前一时刻的输出和source序列两方面的信息，公式如下：**

有了之后，可以在计算**,** 这里计算的是一个条件概率分布，计算出当前情况下target vocab中每个词所对应的概率，这里往往需要一个额外的连接层，因为RNN的输出维度与vocab的维度不同，所以需要经过一个矩阵进行映射。**的计算会使用到，，c三个向量。g函数往往是一个softmax函数，用于计算归一化的概率值。**

1. Seq-to-Seq框架2

第二个要讲的Seq-to-Seq模型来自于“Sequence to Sequence Learning with Neural Networks”，其模型结构如下图：

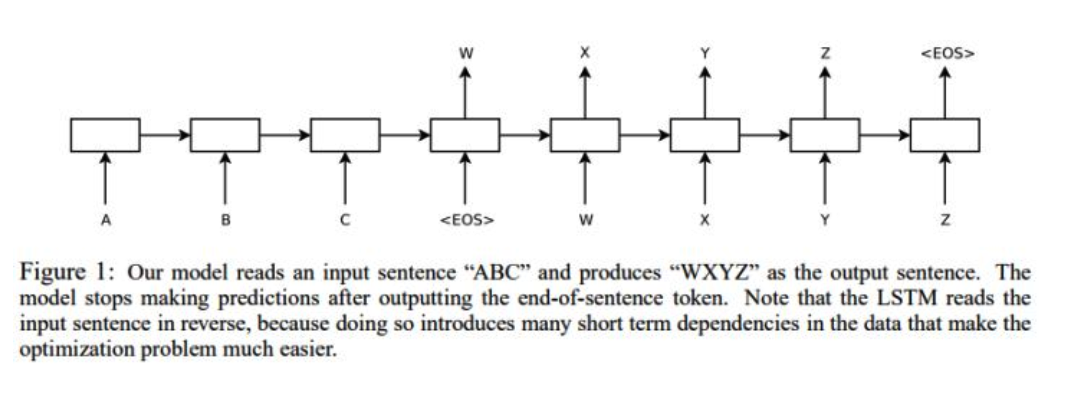


图2：Seq-to-Seq结构

与上面模型的最大区别在于其source编码后的向量c直接作为decoder阶段RNN的初始化状态，而不是在每次decoder时都作为RNN cell的输入。此外，decode时RNN的输入是RNN的前一时刻的输出。首先看下编码阶段：

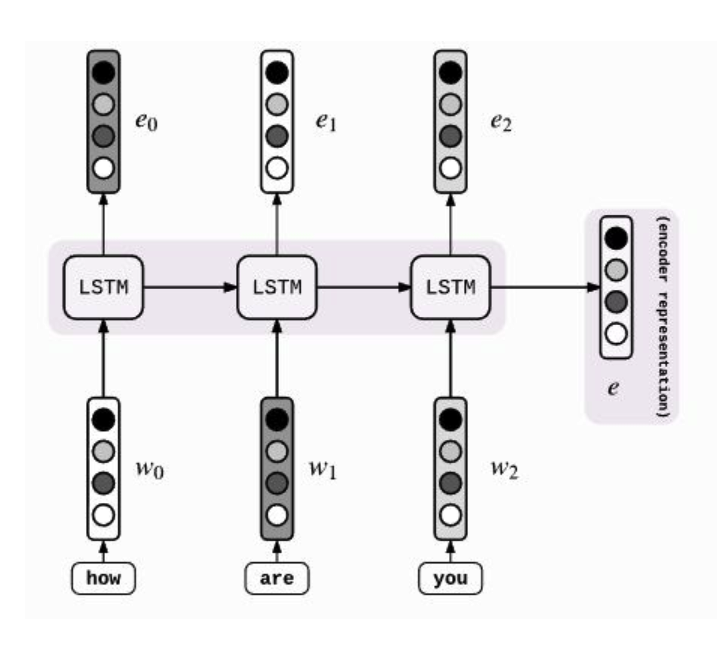


图3：编码阶段

这其实就是简单的RNN模型，每个词经过RNN之后都会编码为hidden state()，并且source序列的编码向量e就是最终的hidden state2。接下来看下解码阶段：

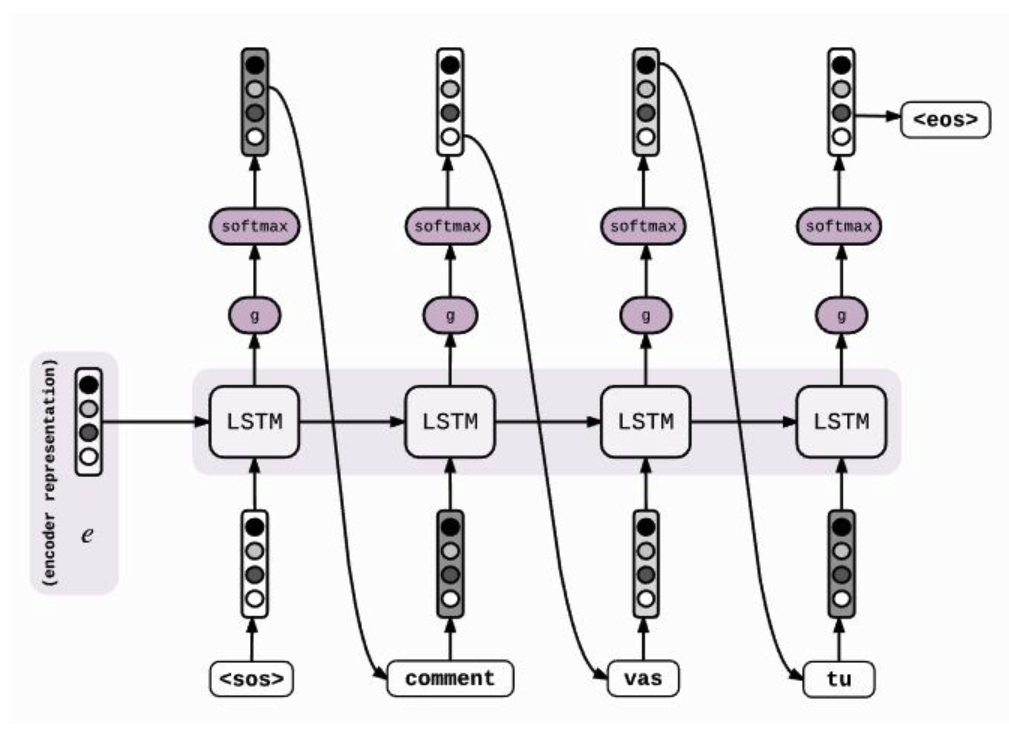


图4：解码阶段

e向量仅作为RNN的初始状态传入decode模型。接下来就是标准的循环神经网络，每一时刻输入都是前一时刻的输出，直到最终的输入<eos>符号停止滚动。

1. Seq-to-Seq with Attention

第三个要讲的是Attention based Seq-to-Seq模型，来自于“Neural Machine Translation by jointly Learning to Align and Translate”这篇论文，其模型结构框架如下：

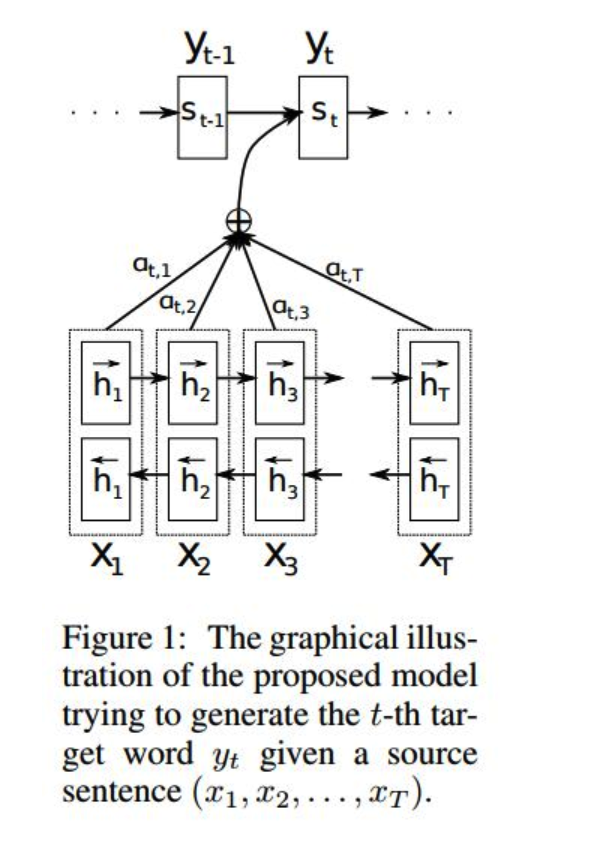


图5：模型结构

前两个模型都是将source序列编码为一个固定维度的向量，但是这样做对于长序列将会丢失很多信息导致效果不好，所以作者提出将encoder阶段所有的隐层状态保存在一个列表中，然后接下来decode的时候，根据前一时刻隐层状态去计算encoder中的T个隐层状态与其相关程度，在进行加权求和后得到编码信息，也就是说每个解码时刻的c向量都是不一样的，会根据target与source之间的相关程度进行权重调整和计算。编码过程就不再赘述了，主要看一下解码过程的流程：

* 1、计算各个隐层向量与之间的相关程度，并进行softmax归一化操作得到每个隐层向量的权重：

其中表示第i个target与第j个隐层向量之间的相关性，可以通过一个MLP神经网络进行计算，在上面示意图中没有显示出来。得到之后，将其传入softmax函数我们就可以得到归一化的权重值。

* 2、对进行加权求和得到此次解码所对应的source编码的向量：