真正的完全图解Seq2Seq Attention模型

原创 盛源车 机器学习算法与自然语言处理 2018-08-03

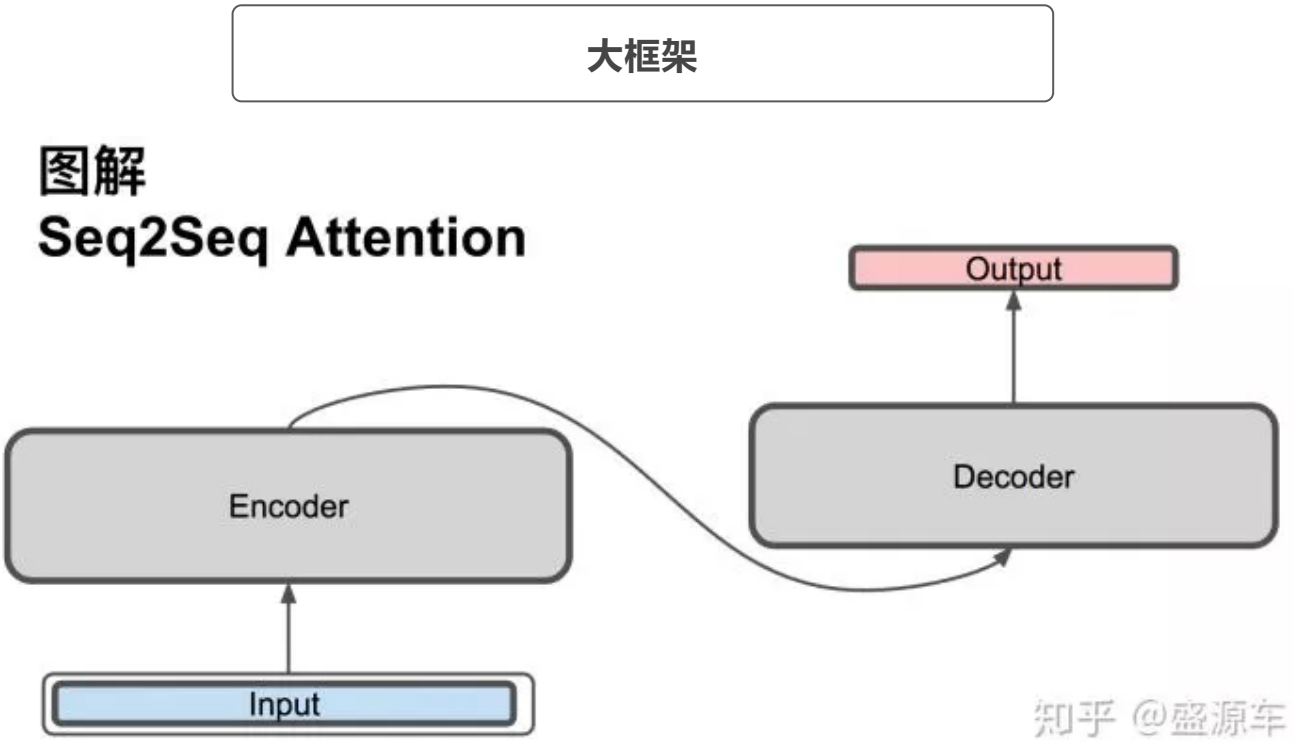
> 作者：盛源车
>
> 知乎专栏：魔法抓的学习笔记

五分钟看懂seq2seq attention模型。

本文通过图片，详细地画出了seq2seq+attention模型的全部流程，帮助小伙伴们**无痛**理解机器翻译等任务的重要模型。

seq2seq 是一个Encoder–Decoder 结构的网络，它的输入**是**一个序列，输出也是一个序列， Encoder 中将一个可变长度的信号序列变为固定长度的向量表达，Decoder 将这个固定长度的向量变成可变长度的目标的信号序列。--简书

好了别管了，接下来开始刷图吧。



想象一下翻译任务，input是一段英文，output是一段中文。

公式（直接跳过看图最佳）

输入： $x = (x_1, \ldots, x_{T_x})$

输出： $y = (y_1, \ldots, y_{T_y})$

(1) $h_t = RNN_{enc}(x_t, h_{t-1})$ ，Encoder方面接受的是每一个单词word embedding，和上一个时间点的hidden state。输出的是这个时间点的hidden state。

(2) $s_t = RNN_{dec}(\hat{y_{t-1}}, s_{t-1})$ ， Decoder方面接受的是目标句子里单词的word embedding，和上一个时间点的hidden state。

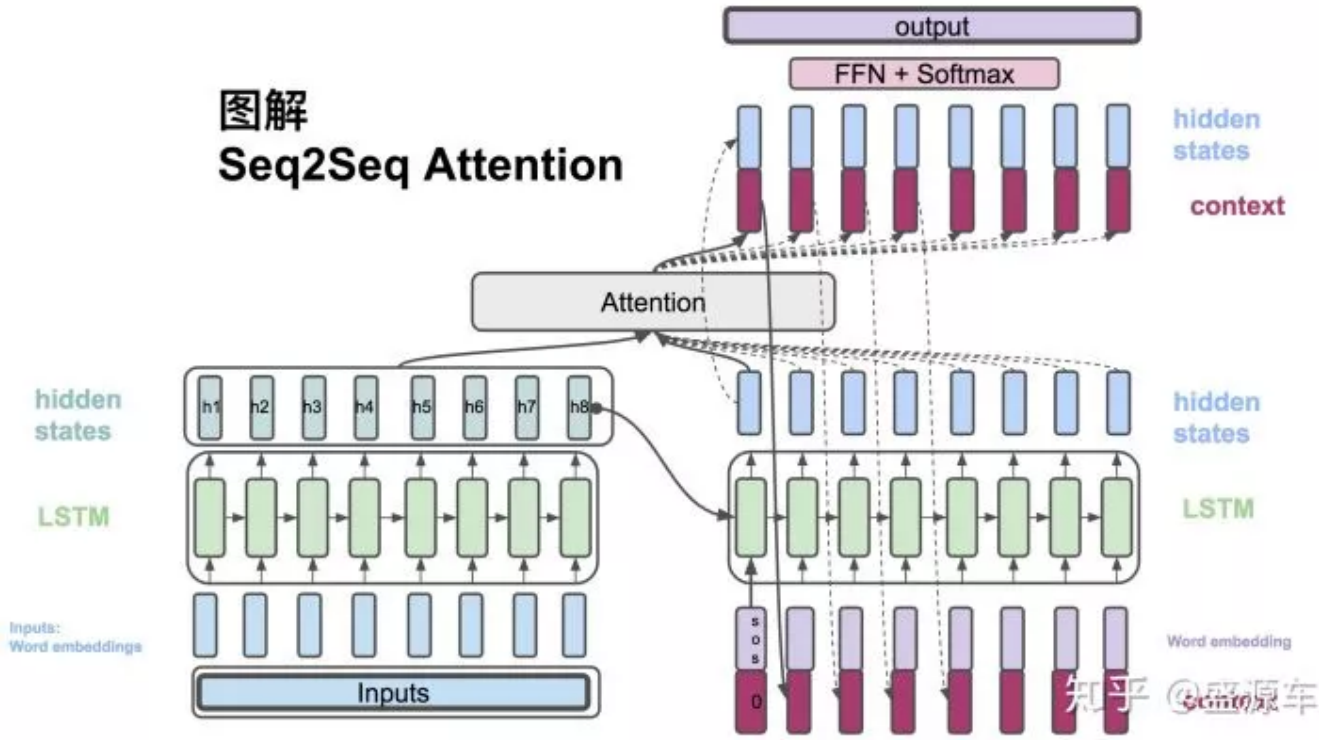
(3) $c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$ ，context vector是一个对于encoder输出的hidden states的一个加权平均。

(4) $\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})}$ ，每一个encoder的hidden states对应的权重。

(5) $e_{ij} = score(s_{i-1}, h_j)$ ，通过decoder的hidden states加上encoder的hidden states来计算一个分数，用于计算权重(4)

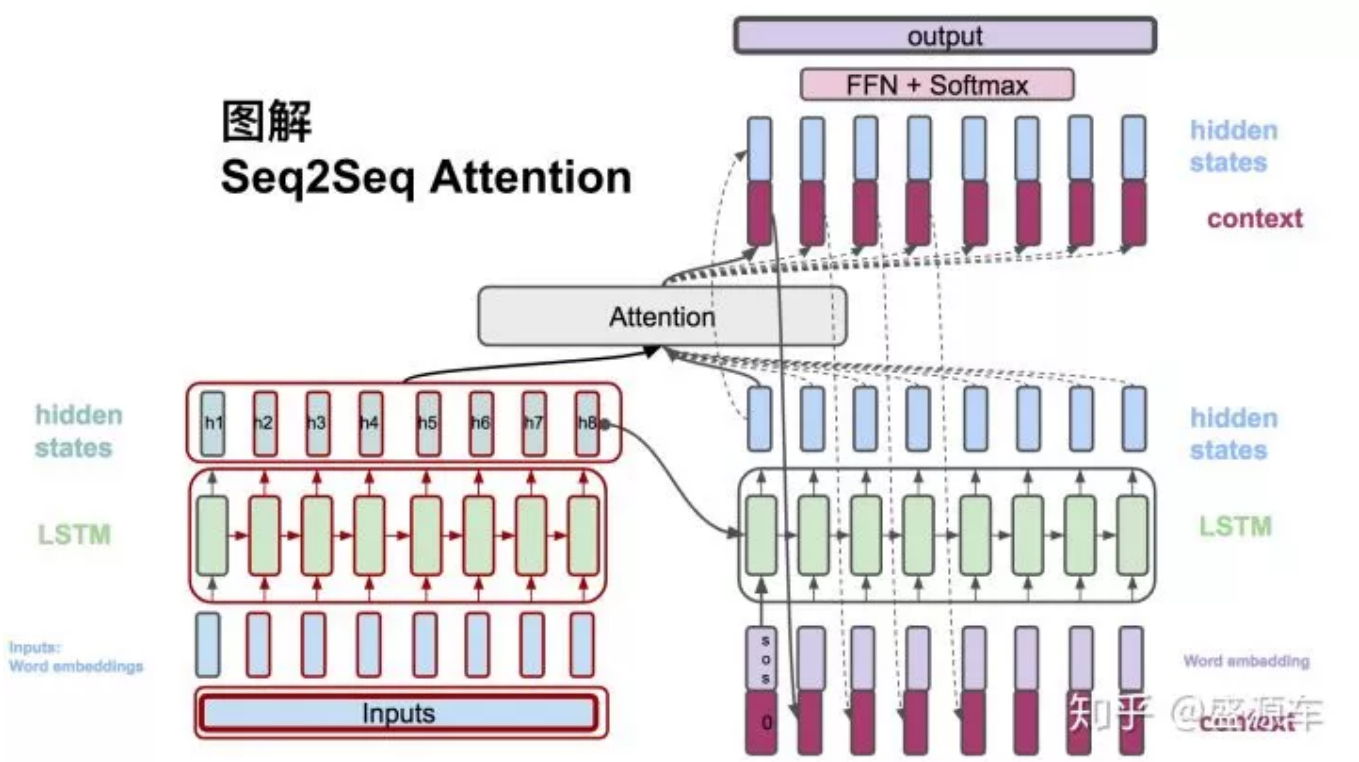
(6) $\hat{s_t} = tanh(W_c[c_t; s_t])$, 将context vector 和 decoder的hidden states 串起来。

详细图



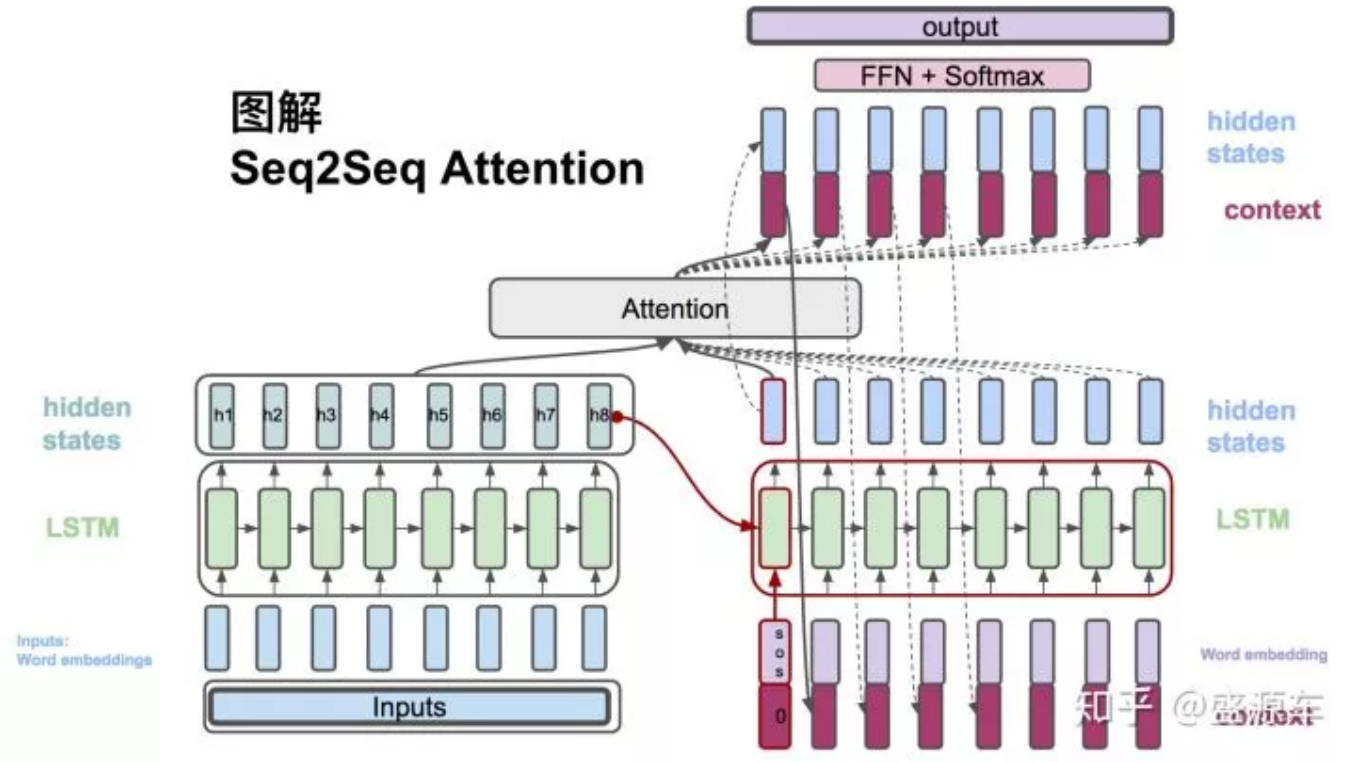
左侧为Encoder+输入，右侧为Decoder+输出。中间为Attention。

(1) $h_t = RNN_{enc}(x_t, h_{t-1})$ ，Encoder方面接受的是每一个单词word embedding，和上一个时间点的hidden state。输出的是这个时间点的hidden state。



从左边Encoder开始，输入转换为word embedding, 进入LSTM。LSTM会在每一个时间点上输出hidden states。如图中的h1,h2,...,h8。

(2) $s_t = RNN_{dec}(\hat{y_{t-1}}, s_{t-1})$ ， Decoder方面接受的是目标句子里单词的word embedding，和上一个时间点的hidden state。

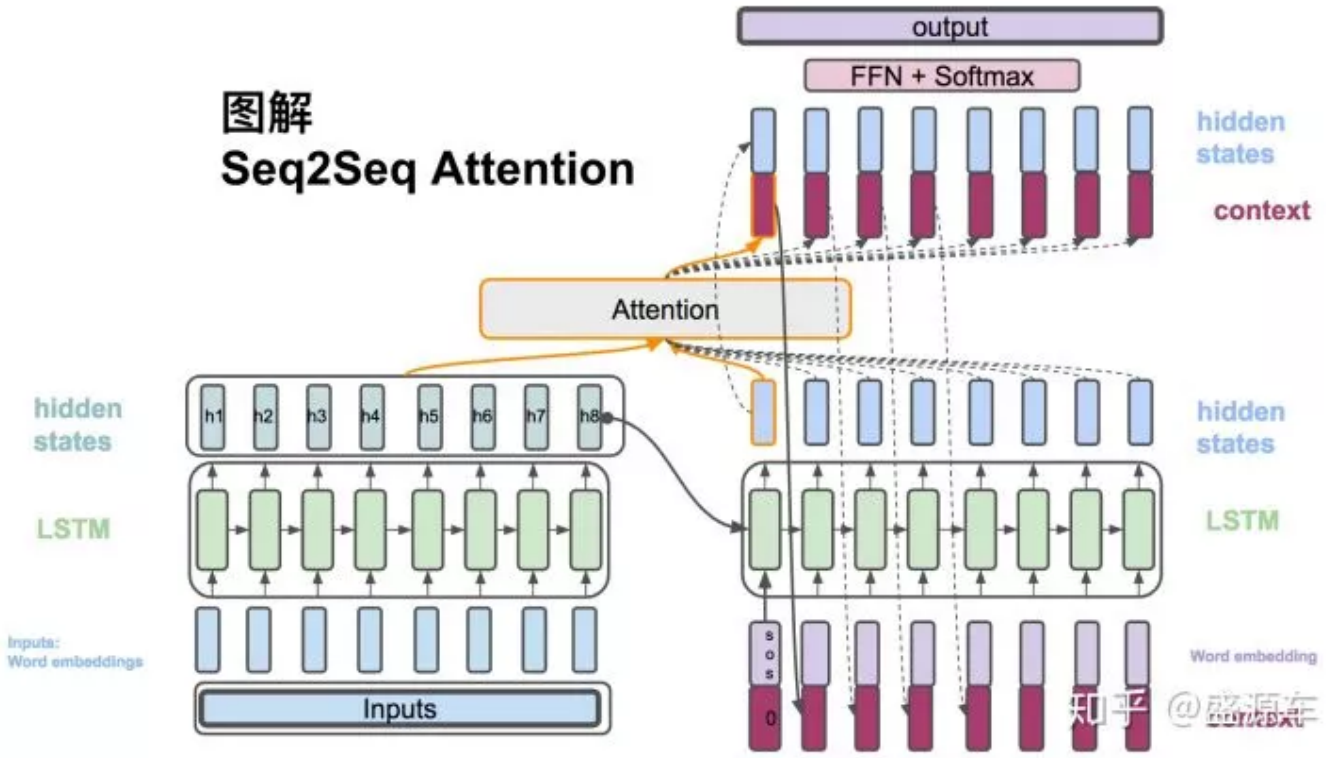


接下来进入右侧Decoder，输入为(1) 句首 <sos>符号，原始context vector(为0)，以及从encoder最后一个hidden state: h8。LSTM的是输出是一个hidden state。（当然还有cell state，这里没用到，不提。）

(3) $c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$ ，context vector是一个对于encoder输出的hidden states的一个加权平均。

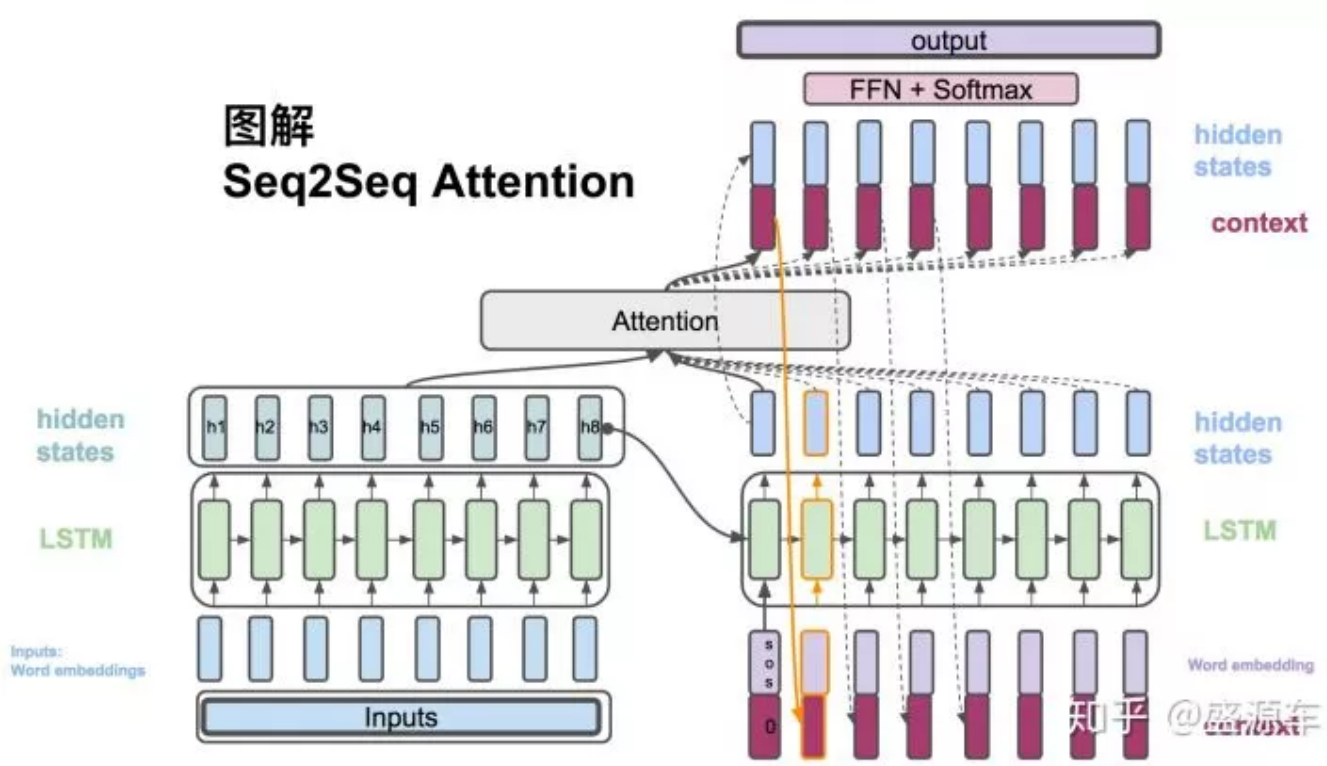
(4) $\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})}$ ，每一个encoder的hidden states对应的权重。

(5) $e_{ij} = score(s_{i-1}, h_j)$ ，通过decoder的hidden states加上encoder的hidden states来计算一个分数，用于计算权重(4)



Decoder的hidden state与Encoder所有的hidden states作为输入，放入Attention模块开始计算一个context vector。之后会介绍attention的计算方法。

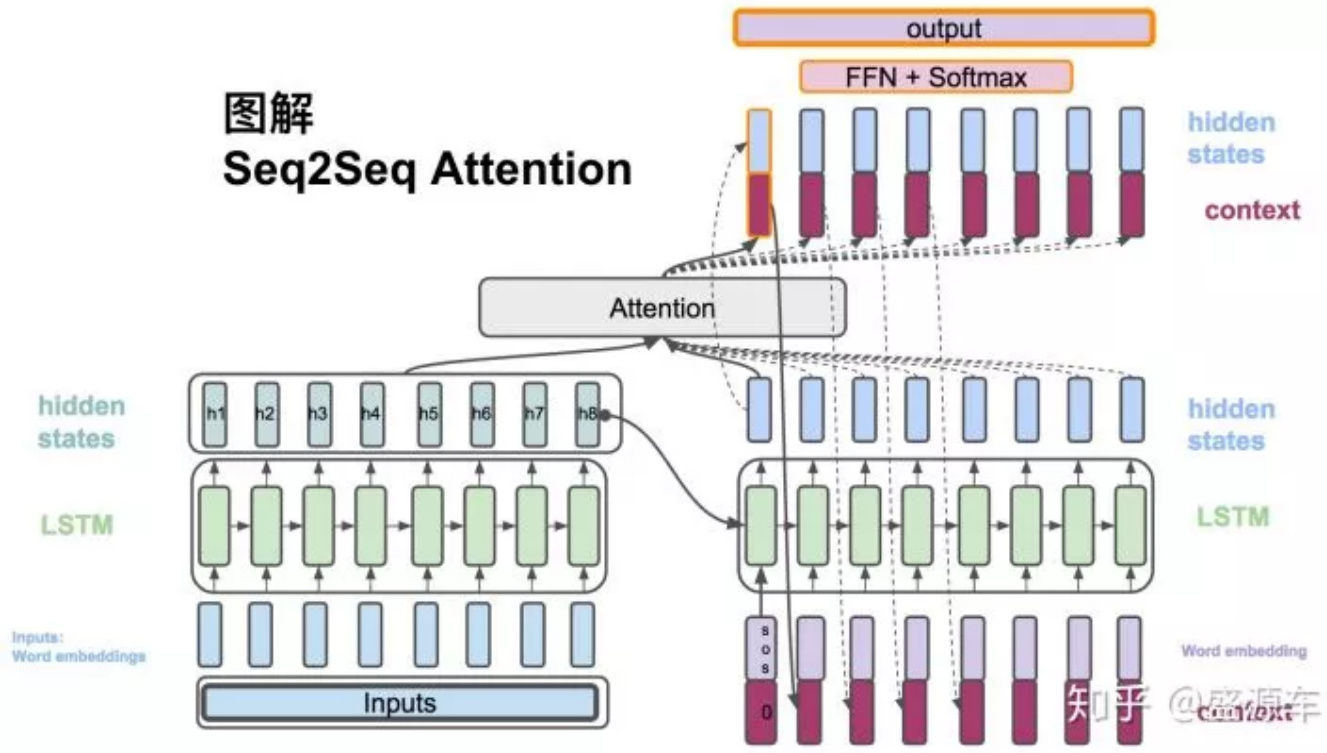
下一个时间点



来到时间点2，之前的context vector可以作为输入和目标的单词串起来作为lstm的输入。之后又回到一个hiddn state。以此循环。

(6) $\hat{s}_t = tanh(W_c[c_t; s_t])$ ，将context vector 和 decoder的hidden states 串起来。

(7) $p(y_t|y_{<t}, x) = softmax(W_s\hat{s}_t)$ ，计算最后的输出概率。

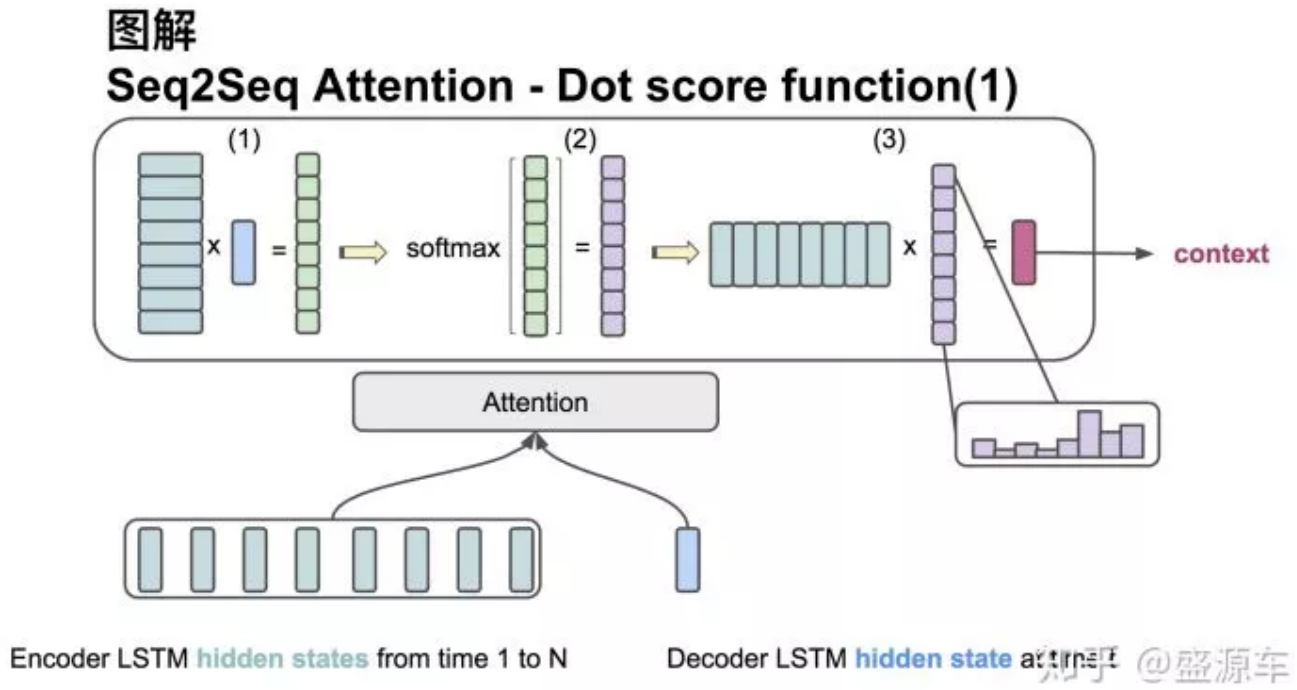


另一方面，context vector和decoder的hidden state合起来通过一系列非线性转换以及softmax最后计算出概率。

在luong中提到了三种score的计算方法。这里图解前两种：

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W}_a \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W}_a [\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s]\right) & \textit{concat} \end{cases}$$

Attention score function: **dot**



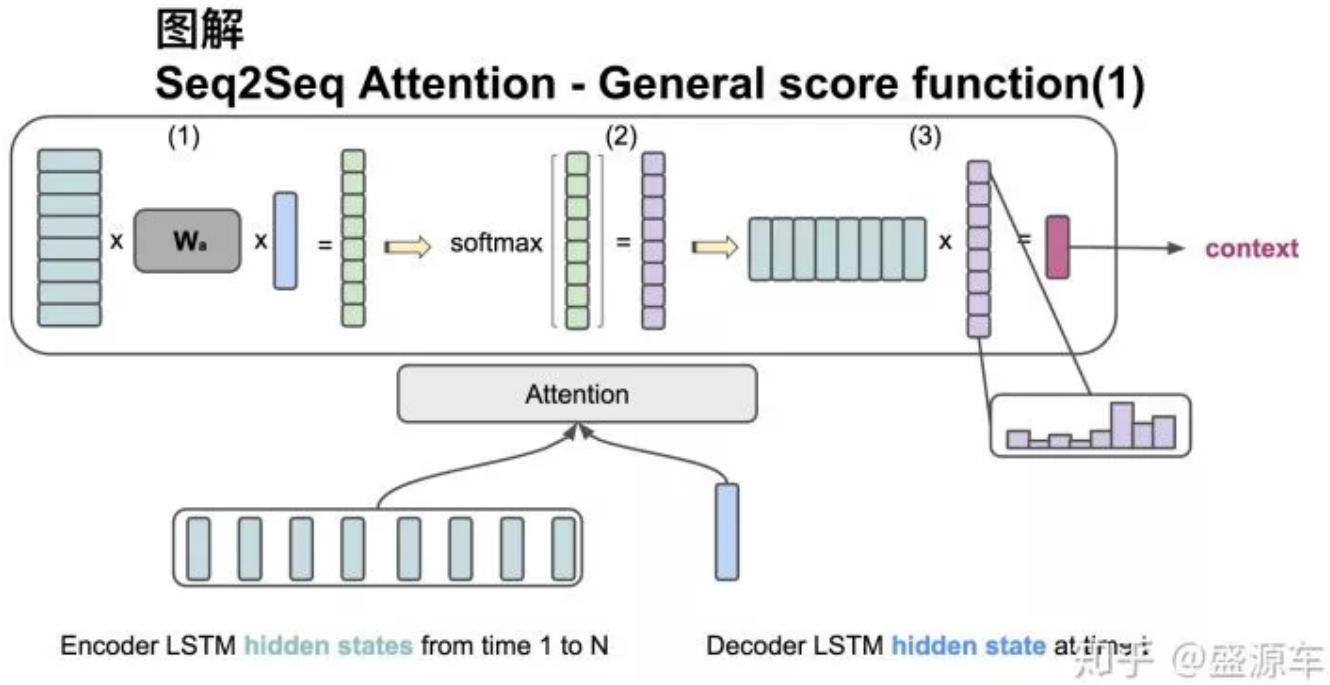
输入是encoder的所有hidden states H: 大小为(hid dim, sequence length)。decoder在一个时间点上的hidden state， s： 大小为（hid dim, 1）。

第一步：旋转H为（sequence length, hid dim) 与s做点乘得到一个 大小为(sequence length, 1)的**分数。**

第二步：对分数做softmax得到一个合为1的**权重。**

第三步：将H与第二步得到的权重做点乘得到一个大小为(hid dim, 1)的**context vector。**

Attention score function: **general**



输入是encoder的所有hidden states H: 大小为(**hid dim1**, sequence length)。decoder在一个时间点上的hidden state， s： 大小为（**hid dim2**, 1）。此处两个hidden state的*纬度并不一样。*

第一步：旋转H为（sequence length, hid dim1) 与 **Wa [大小为 hid dim1, hid dim 2)]** 做点乘， 再和s做点乘得到一个 大小为(sequence length, 1)的**分数。**

第二步：对分数做softmax得到一个合为1的**权重。**

第三步：将H与第二步得到的权重做点乘得到一个大小为(hid dim, 1)的**context vector。**

完结

看懂一个模型的最好办法就是在心里想一遍从输入到模型到输出每一个步骤里，tensor是如何流动的。

END

推荐阅读：

闲聊结构化预测（structured learning）【这是一类问题】

我就不信看完这篇你还搞不懂信息熵