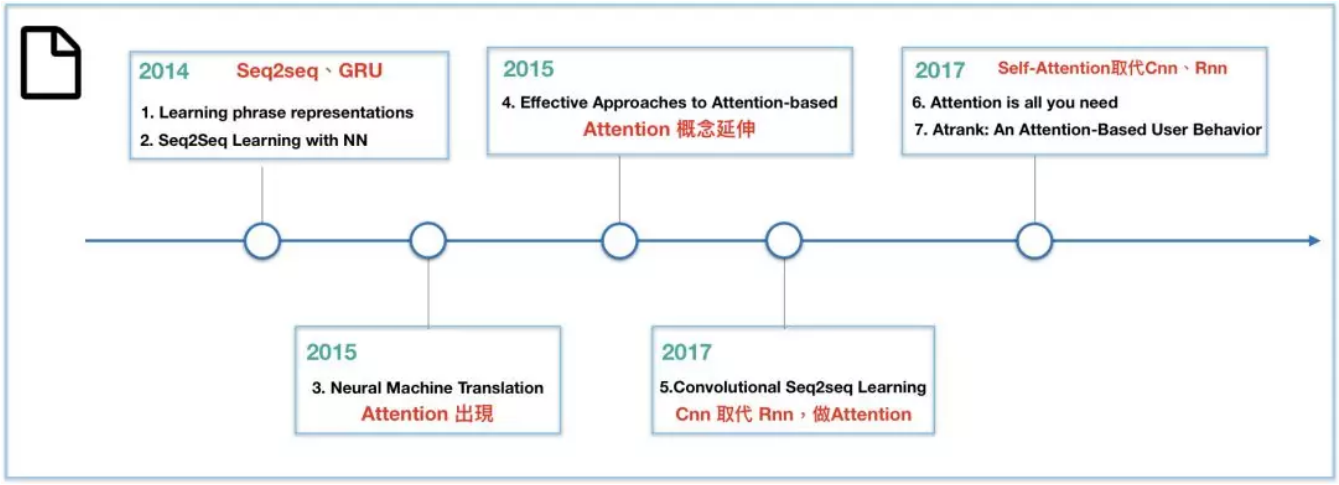
从Seq2seq到Attention模型到Self Attention（一）

原创 QIML编辑部 量化投资与机器学习 2018-10-08

——作者：Bgg——

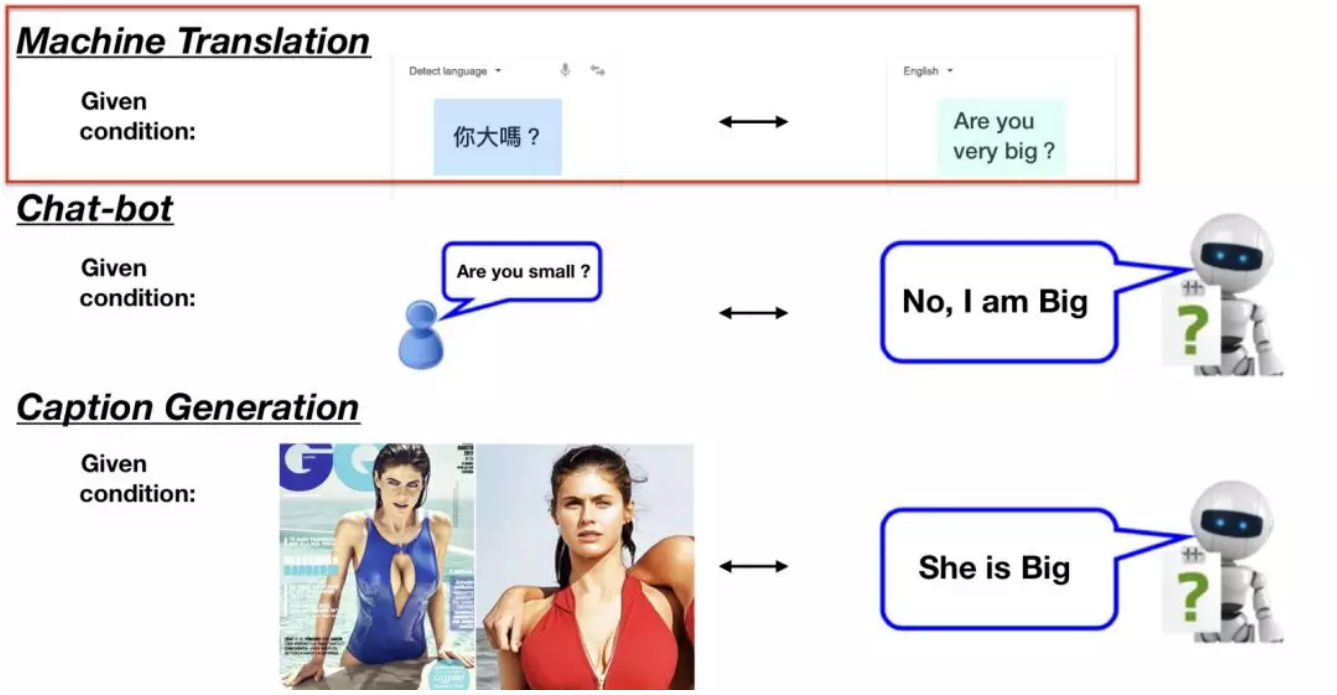
近一两年，注意力模型（Attention Model）是深度学习领域最受瞩目的新星，用来处理与序列相关的数据，特别是2017年Google提出后，模型成效、复杂度又取得了更大的进展。以金融业为例，客户的行为代表一连串的序列，但要从串行化的客户历程数据去萃取信息是非常困难的，如果能够将self-attention的概念应用在客户历程并拆解分析，就能探索客户潜在行为背后无限的商机。然而，笔者从Attention model读到self attention时，遇到不少障碍，其中很大部分是后者在论文提出的概念，鲜少有文章解释如何和前者做关联，笔者希望藉由这系列文，解释在机器翻译的领域中，是如何从Seq2seq演进至Attention model再至self attention，使读者在理解Attention机制不再这么困难。

为此，系列文分为两篇，第一篇着重在解释Seq2seq、Attention模型，第二篇重点摆在self attention，希望大家看完后能有所收获。



前言

你可能很常听到Seq2seq这词，却不明白是什么意思。Seq2seq全名是Sequence-to-sequence，也就是从序列到序列的过程，是近年当红的模型之一。Seq2seq被广泛应用在机器翻译、聊天机器人甚至是图像生成文字等情境。如下图：

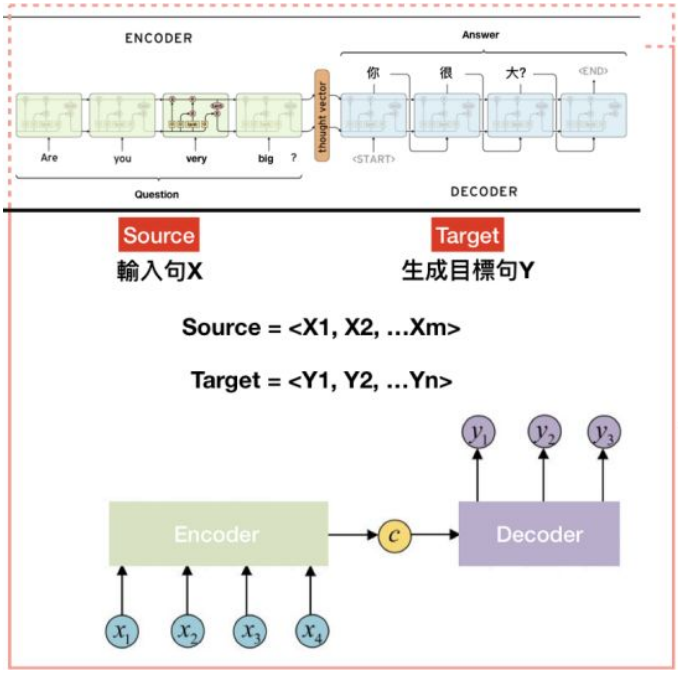


其中，Seq2seq常见情境为机器翻译，因此接下来的内容都会以情境进行说明。

图（3）是个典型的Seq2seq模型，包含了编码器（Encoder）和解码器（Decoder）.只要输入句子至Encoder，即可从Decoder获得目标句。

举例来说，如果我们将“Are you very big”作为输入句（source sentence），即可得到目标句（target sentence）“你很大？”。机器翻译就是这么简单，然而，如果想了解它如何组成，会发现其中充斥着各种难以咀嚼的RNN/LSTM等概念。

接下来，让我们快速回味一下RNN/LSTM，方便后续模型理解。



Decoder Formula

Hence, the hidden state of the decoder at time t is computed by,

$$\mathbf{h}_{\langle t \rangle} = f\left(\mathbf{h}_{\langle t-1 \rangle}, y_{t-1}, \mathbf{c}\right),$$

and similarly, the conditional distribution of the next symbol is

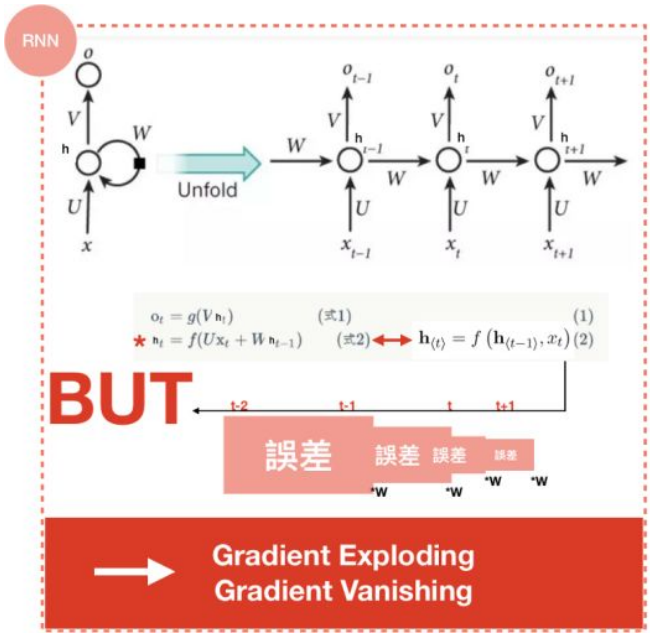
$$P(y_t | y_{t-1}, y_{t-2}, \ldots, y_1, \mathbf{c}) = g\left(\mathbf{h}_{\langle t \rangle}, y_{t-1}, \mathbf{c}\right).$$

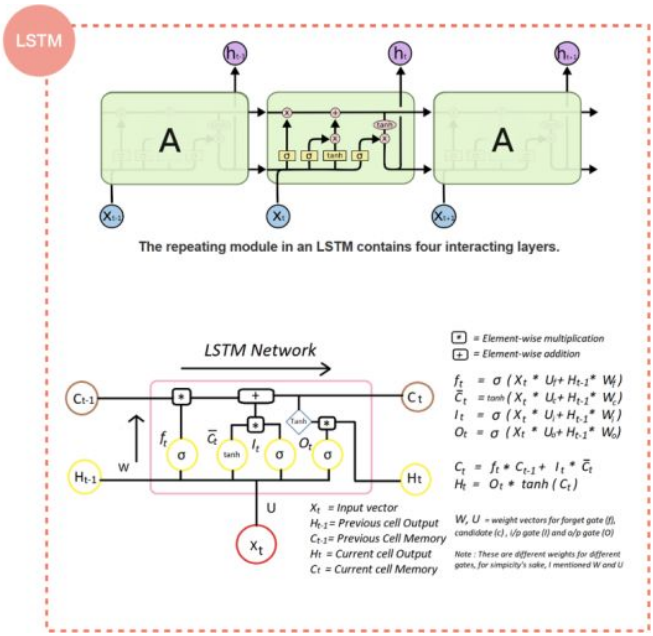
for given activation functions f and g (the latter must produce valid probabilities, e.g. with a softmax).

RNN/LSTM

RNN是DNN模型的变种，不同之处在于它可以储存过去的行为记忆，进行更准确的预测，然而，就像人脑一样，一旦所需记忆量太大，就会比较健忘。我们可以把隐藏状态（hidden state）h_{t}认为是记忆单元，h_{t}可通过前一步的hidden state和当前时刻的输入（input）得到，因为是记忆单元，h_{t}可以捕捉到之前所有时刻产生的信息，而输出（output）o_{t}仅依赖于t时刻的记忆，也就是h_{t}。

RNN在反向训练误差时，都会乘上参数，参数乘上误差的结果，大则出现梯度爆炸；小则梯度消失，导致模型成效不佳，如图4。

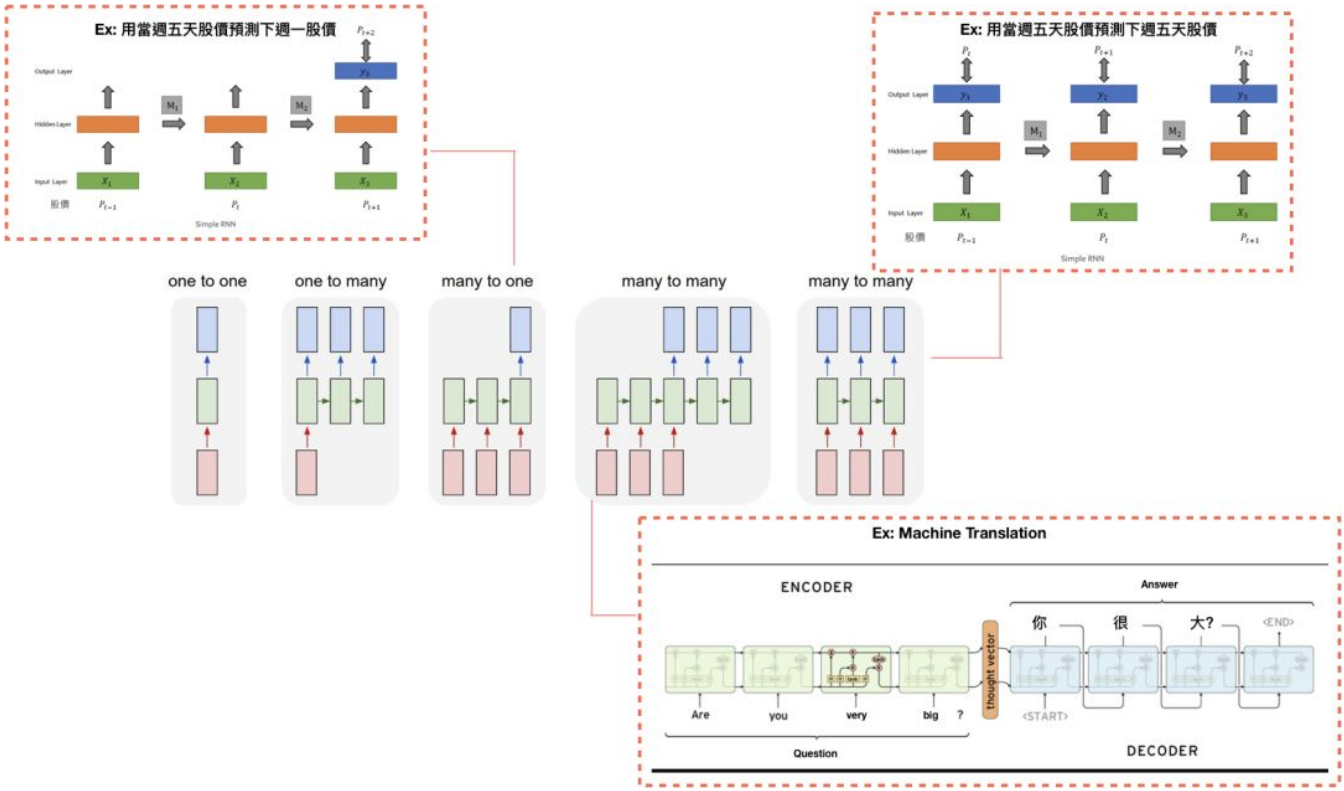


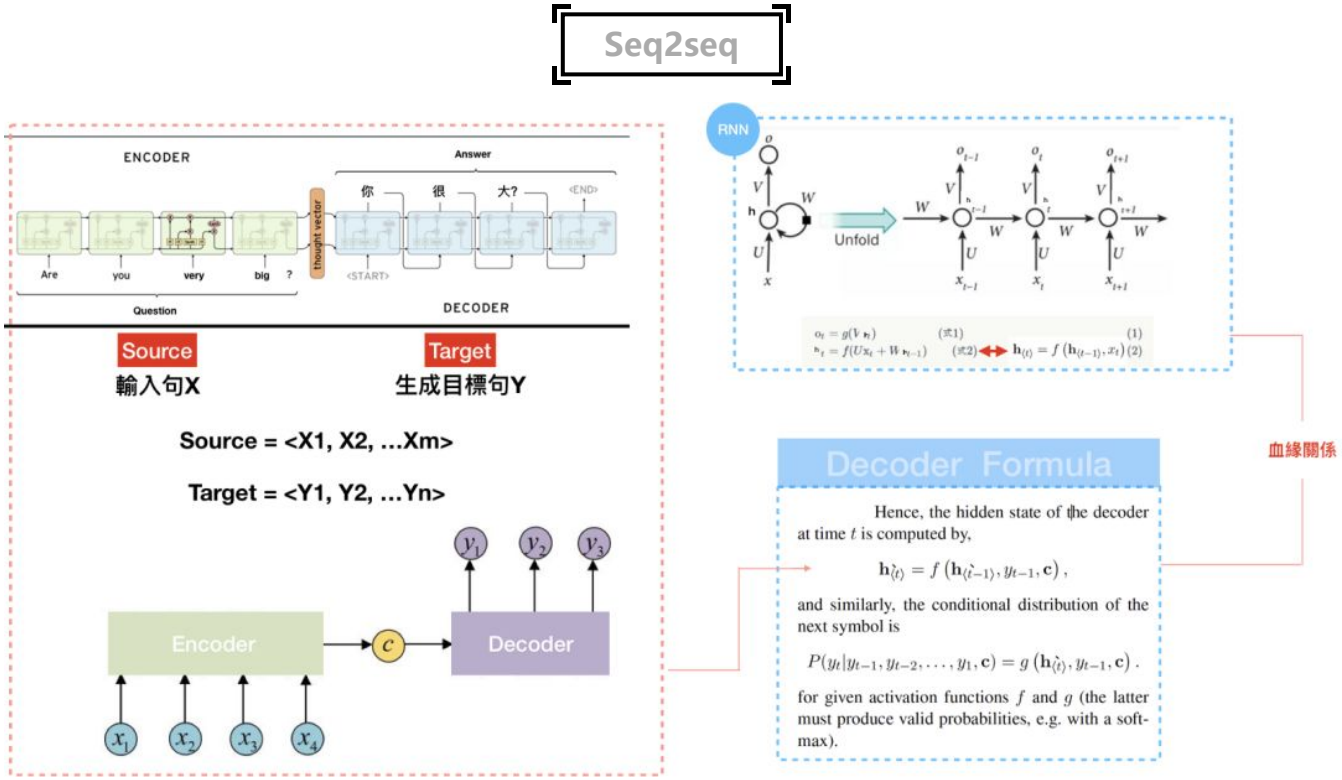


为了解决健忘、训练误差的问题，LSTM有了像是遗忘/输入/输出门（forget/input/output gate），隐藏状态（hidden state），记忆单元（cell memory）等概念，带来了更好的结果。在2014年，论文

Learning Phrase Representations除了提出Seq2seq的概念，更提出了LSTM的简化版GRU，此后，LSTM和GRU便取代RNN成为深度学习当中的主流。

下图是LSTM的各种应用，在此不深入描述。

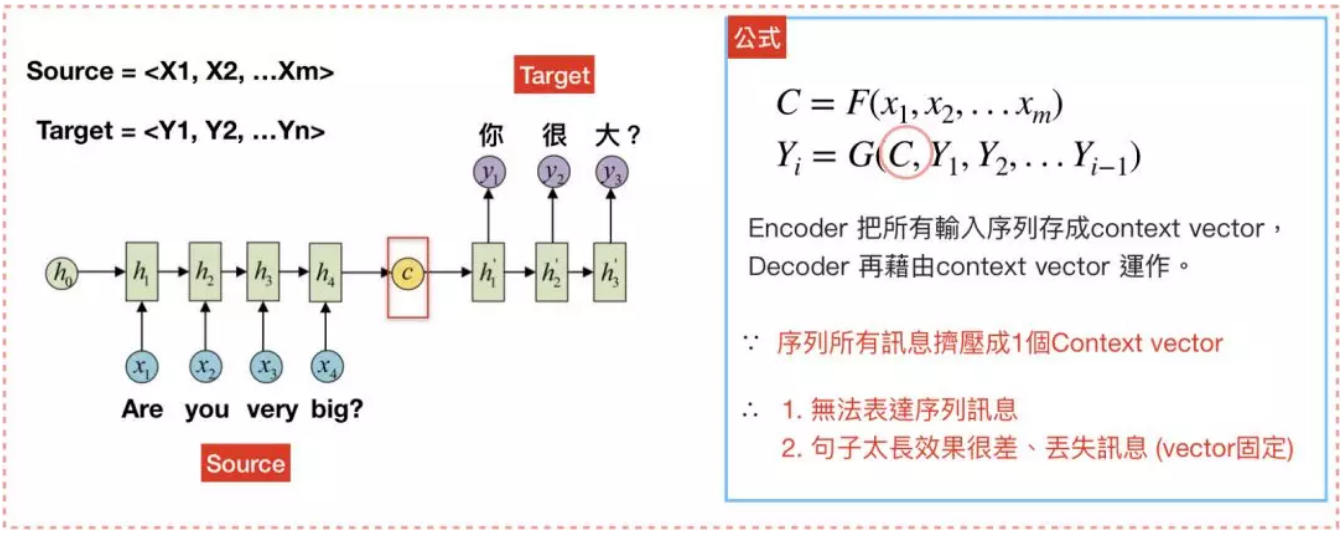




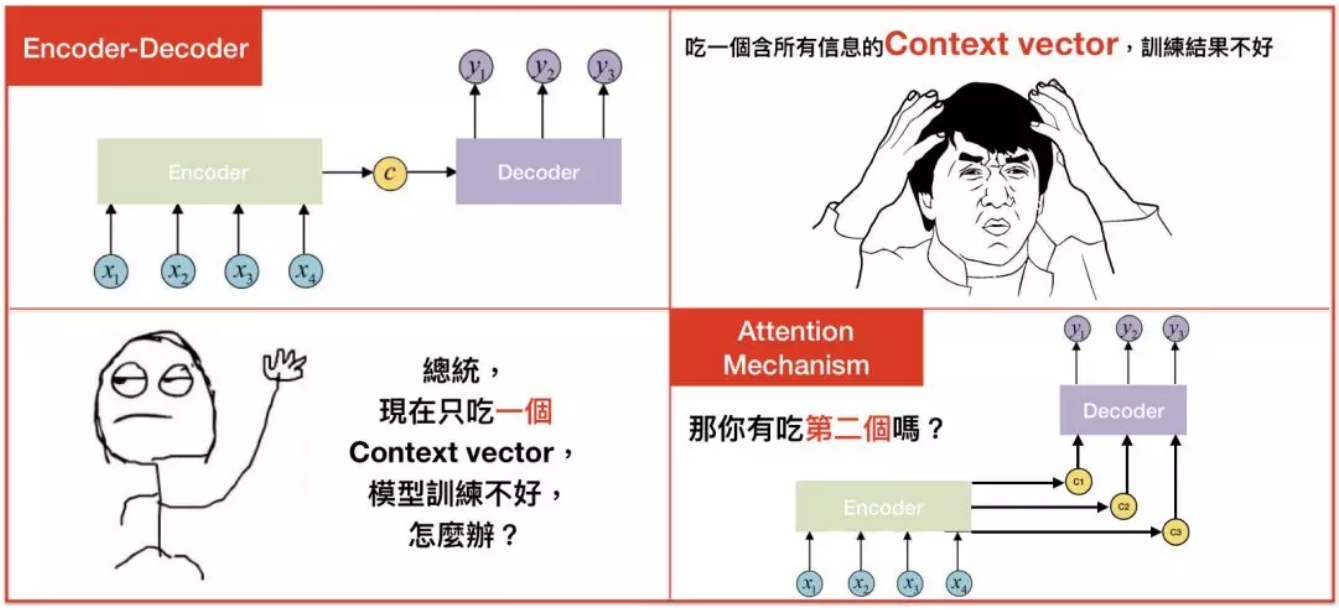
回到正题，所以Seq2seq是怎么组成的？我们可以看到Seq2seq包含两部分：Encoder和Decoder。一旦将句子输入至Encoder，即可从Decoder获得目标句。本篇文章着墨在Decoder生成过程，Encoder

就是个单纯的RNN/ LSTM，读者若有兴趣可再自行研究，此外RNN/LSTM可以互相代替，以下仅以RNN作为解释。

现在我们具备RNN/LSTM的知识，可以发现Seq2seq中，Decoder的公式和RNN根本就是同一个模子出来的，差别在于Decoder多了一个C — 图（6），这个C是指context vector/thought vector。context vector可以想成是一个含有所有输入句信息的向量，也就是Encoder当中，最后一个hidden state。简单来说，Encoder将输入句压缩成固定长度的context vector，context vector即可完整表达输入句，再透过Decoder将context vector内的信息产生输出句，如图7。



但是，在Seq2seq模型中，Encoder将输入句压缩成固定长度的context vector真的好吗？如果句子今天很长，固定长度的context vector效果就会不好。怎么办呢？



在2015年，有个救星诞生了，叫作注意力模型（attention model）。

Attention model

为什么要用attention model?

The attention model用来帮助解决机器翻译在句子过长时效果不佳的问题。

这种新的构架替输入句的每个文字都创造一个context vector，而非仅仅替输入句创造一个从最终的hidden state得来的context vector，举例来说，如果一个输入句有N个文字，就会产生N个context vector，好处是，每个context vector能够被更有效的译码。

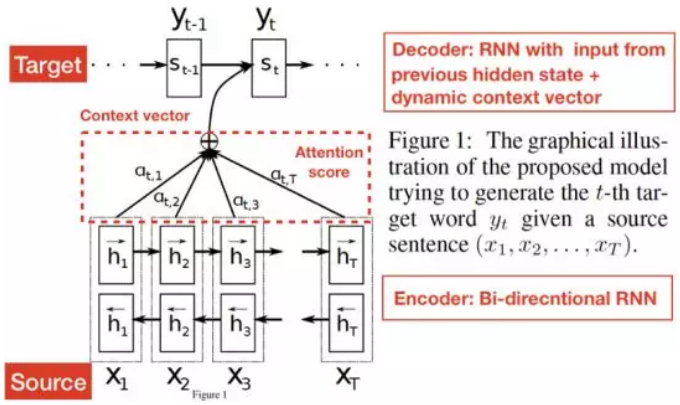
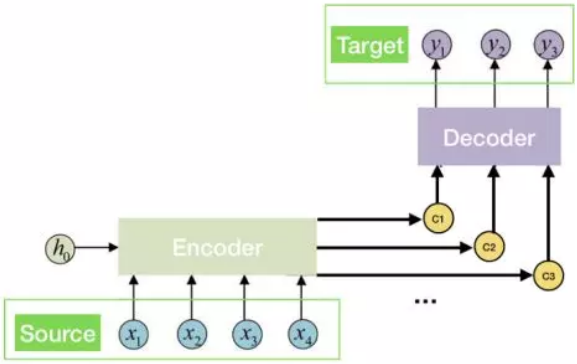


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.



在Attention model中，Encoder和Seq2seq概念一样，一样是从输入句<X1，X2，X3...Xm>产生<h1，h2，h....hm>的hidden state，再计算目标句<y1...yn>。换言之，就是将输入句作为input而目标句作为output，所以差别就在于context vector c_{i}是怎么计算?

Decoder Formula

Target word $p(y_i|y_1, \ldots, y_{i-1}, \mathbf{x}) = g(y_{i-1}, s_i, c_i)$.

where s_i is an RNN hidden state for time i, computed by

$$s_i = f(s_{i-1}, y_{i-1}, c_i).$$

The context vector c_i is, then, computed as a weighted sum of these annotations h_i:

i個Context vector $c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j.$ (5)

The weight α_{ij} of each annotation h_j is computed by

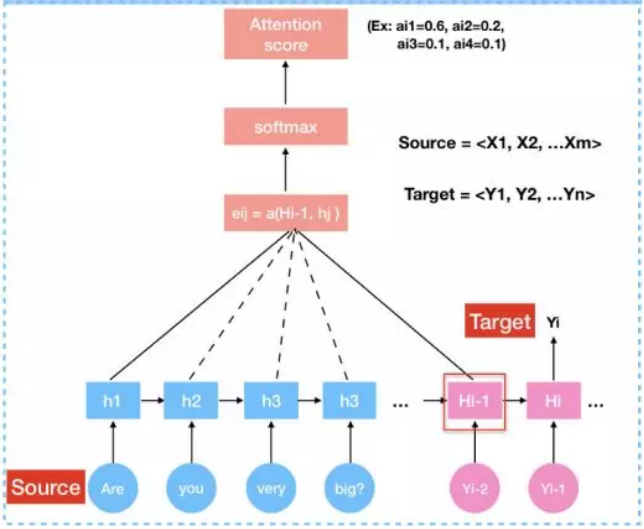
Attention score $\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$, (6)

where

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

is an *alignment model* which scores how well the inputs around position j and the output at position i match. The score is based on the RNN hidden state s_{i-1} (just before emitting y_i, Eq. (4)) and the j-th annotation h_j of the input sentence.

Brief picture of Attention score



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

Context vector c_{i}是透过attention scoreα乘上input的序列加权求和.Attention/Alignment score是attention model中提出一个很重要的概念，可以用来衡量输入句中的每个文字对目标句中的每个文字所带来重要性的程度。由公式可知，attention score藉由score e_{ij}所计算得到，所以先来看看score e_{ij}是什么。

$$\underline{Attention\ score}\ \alpha_{ij} = e^{e_{ij}} / \sum_{k=1}^{T_x} e^{e_{ik}}$$

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

在计算score中，a代表Alignment model会根据输入字位置j和输出字位置i这两者的关联程度，计算出一个score e_{ij}。换言之，e_{i，j}是衡量RNN decoder中的hidden state s_{i-1}和输入句中的第j个文字hidden state h_{j}的关系所计算出的权重—如方程式3，那权重怎么算呢?

3.1 Global Attention

The idea of a global attentional model is to consider all the hidden states of the encoder when deriving the context vector c_t. In this model type, a variable-length alignment vector $\boldsymbol{a}_t$, whose size equals the number of time steps on the source side, is derived by comparing the current target hidden state $\boldsymbol{h}_t$ with each source hidden state $\bar{\boldsymbol{h}}_s$:

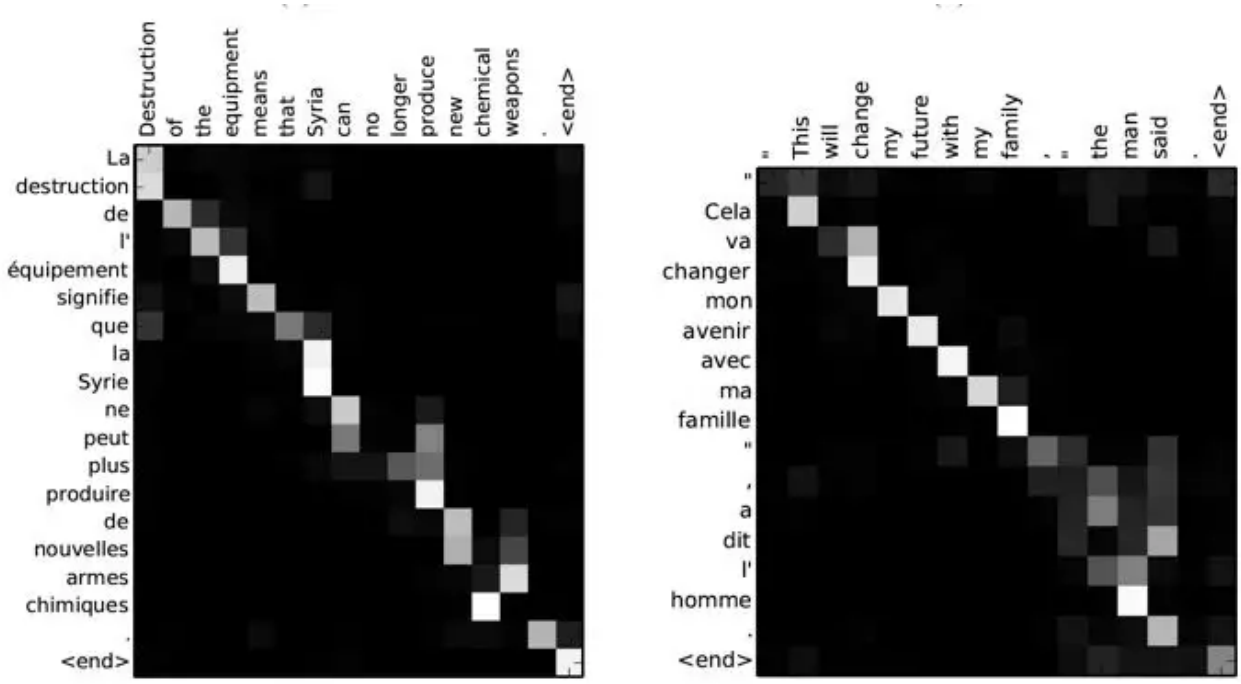
$$\begin{aligned} \boldsymbol{a}_t(s) &= \text{align}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) \qquad (7) \\ &= \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)} \end{aligned}$$

Here, score is referred as a *content-based* function for which we consider three different alternatives:

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W}_{\boldsymbol{a}} \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{W}_{\boldsymbol{a}}[\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s] & \textit{concat} \end{cases} \qquad (8)$$

Neural Machine Translation发表之后，接续的论文Effective approaches of the NMT、Show，Attend and Tell提出了global/local attention和soft/hard attention的概念，而score e_{ij}的计算方式类似global和soft attention。细节在此不多说，图11可以看到3种计算权重的方式，我们把刚才的公式做些改变，将score e_{ij}改写成score（h_{t}，\bar {h_{s}}），h_{t}代表s_{i-1}而\bar {h_{s}}代表h_{j}，为了计算方便，我们采用内积（dot）计算权重。

有了score e_{ij}，即可透过softmax算出attention score，context vector也可得到，在attention model中，context vector又称为attention vector。我们可以将attention score列为矩阵，透过此矩阵可看到输入端文字和输出端文字间的对应关系，也就是论文当中提出align的概念。

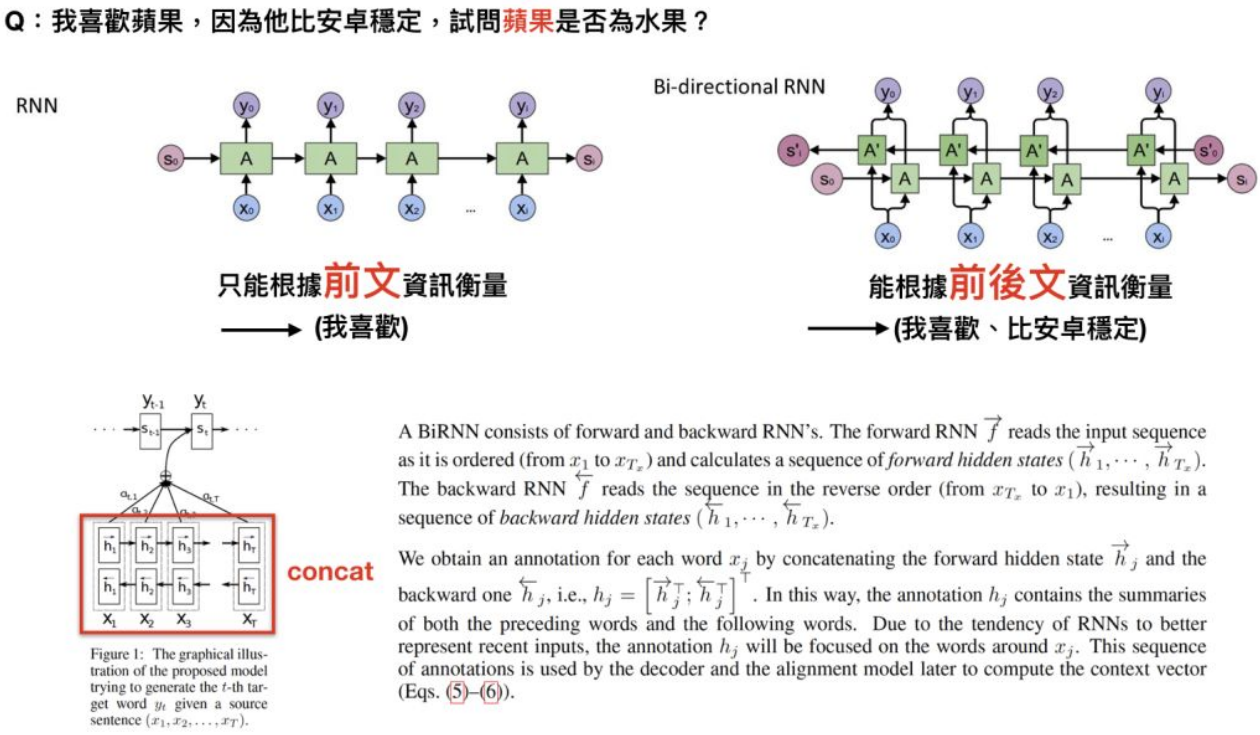


(c) (d)

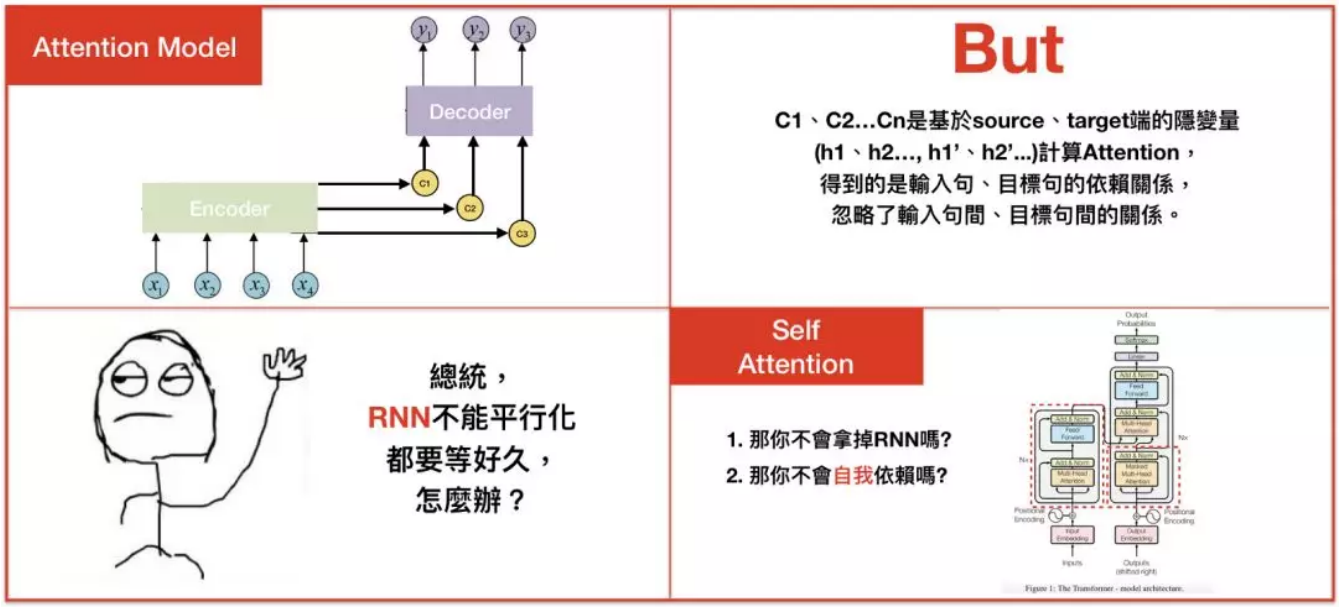
我们知道如何计算context vector后，回头看encoder。

attention model中的encoder用的是改良版RNN：双向RNN（Bi-directional RNN），以往单向RNN的问题在于t时刻时，只能透过之前的信息进行预测，但事实上，模型有时候可能也需要利用未来时刻的信息进行预测，其运作模式为，一个hidden layer用来由左到右，另一个由右到左，透过双向RNN，我们可以对词语进行更好的预测。

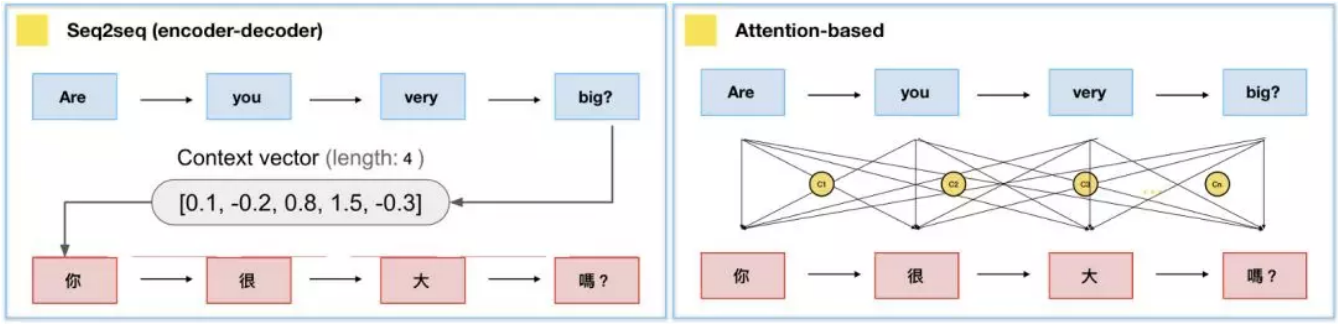
举例来说，“我喜欢苹果，因为它很好吃”？和“我喜欢苹果，因为他比安卓稳定”这两个句子当中，如果只看“我喜欢苹果”，你可能不知道苹果指的是水果还是手机，但如果可以根据后面那句得到信息，答案就很显而易见，这就是双向RNN运作的方式。



Attention model虽然解决了输入句仅有一个context vector的缺点，但依旧存在不少问题。1.context vector计算的是输入句、目标句间的关联，却忽略了输入句中文字间的关联，和目标句中文字间的关联性，2.不管是Seq2seq或是Attention model，其中使用的都是RNN，RNN的缺点就是无法平行化处理，导致模型训练的时间很长，有些论文尝试用CNN去解决这样的问题，像是Facebook提出的Convolutional Seq2seq learning，但CNN实际上是透过大量的layer去解决局部信息的问题，在2017年，Google提出了一种叫做"The transformer"的模型，透过self attention、multi-head的概念去解决上述缺点，完全舍弃了RNN、CNN的构架。



让我们复习一下Seq2seq、Attention model，差别在于计算context vector的方式。



总结

透过上述内容，我们快速的了解Seq2seq、Attention model运作、计算方式，我强烈建议有兴趣的读者可以参考图1中的论文，会有很多收获。

系列二将着重在Google于论文"Attention is all you need"所提出的self attention、multi-head等概念。

参考

[1] Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translationr. arXiv:1406.1078v3 (2014).

[2] Sequence to Sequence Learning with Neural Networks. arXiv:1409.3215v3 (2014).

[3] Neural machine translation by joint learning to align and translate. arXiv:1409.0473v7 (2016).

[4] Effective Approaches to Attention-based Neural Machine Translation. arXiv:1508.0402v5 (2015).

[5] Convolutional Sequence to Sequence learning. arXiv:1705.03122v3(2017).

[6] Attention Is All You Need. arXiv:1706.03762v5 (2017).

[7] ATRank: An Attention-Based User Behavior Modeling Framework for Recommendation. arXiv:1711.06632v2 (2017).

[8] Key-Value Memory Networks for Directly Reading Documents. arXiv:1606.03126v2 (2016).

[9] Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. arXiv:1502.03044v3 (2016).

[10] Deep Residual Learning for Image Recognition. arXiv:1512.03385v1 (2015).

[11] Layer Normalization. arXiv:1607.06450v1 (2016).

来源：

https://medium.com/@bgg/seq2seq-pay-attention-to-self-attention-part-1-d332e85e9aad

推荐阅读