

Machine Translation

循环神经网络 (RNN)

对于处理输入、输出不定长且存在上下文依赖的序列数据，类似DNN、CNN网络其效率较低，且无法解决依赖问题。对此我们需引入循环神经网络。（RNN， Recurrent Neural Network）（区别递归神经网络 RNN， Recursive Neural Network。循环神经网络可以看做是数据以链状结构展开，而递归神经网络数据则以树状结构展开，其主要刻画数据或知识间的推理过程，然而树的构建仍是目前的难点）RNN的核心思想即是将数据按时间轴展开，每一时刻数据均对应相同的神经单元，且上一时刻的结果能传递至下一时刻。至此便解决了输入输出变长且存在上下文依赖的问题。

循环神经网络可以看做在做“加法”操作，即通过加法中的进位操作，将上一时刻的信息或结果传递至下一时刻，如下所示：

三位加法单元

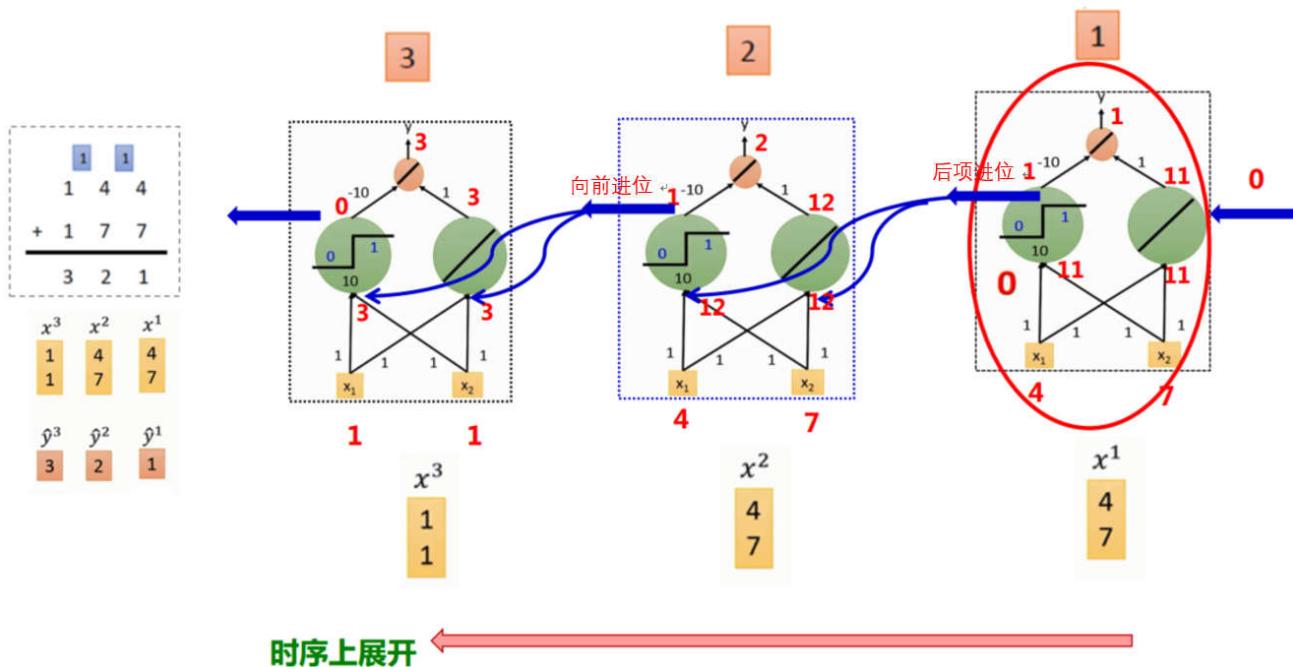


图1. 加法单元

RNN单元结构及其在时序上的展开如下图所示：

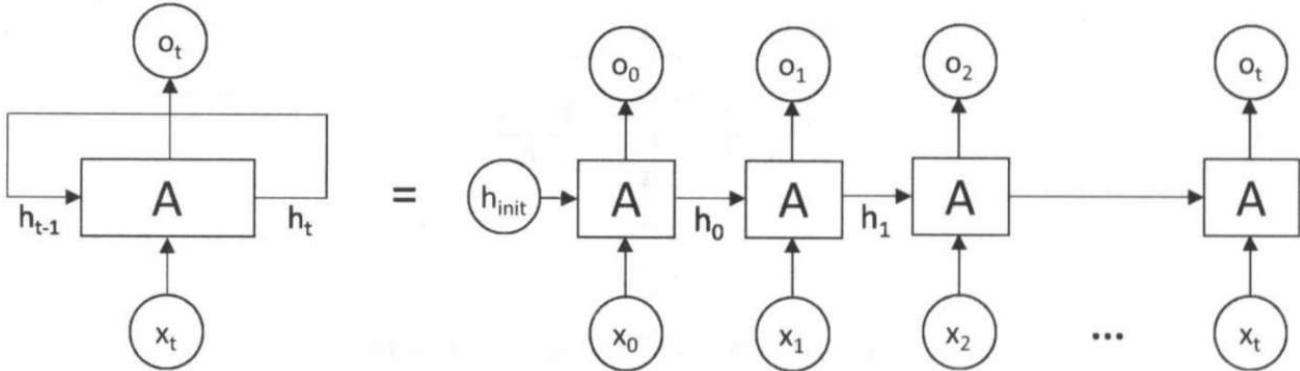
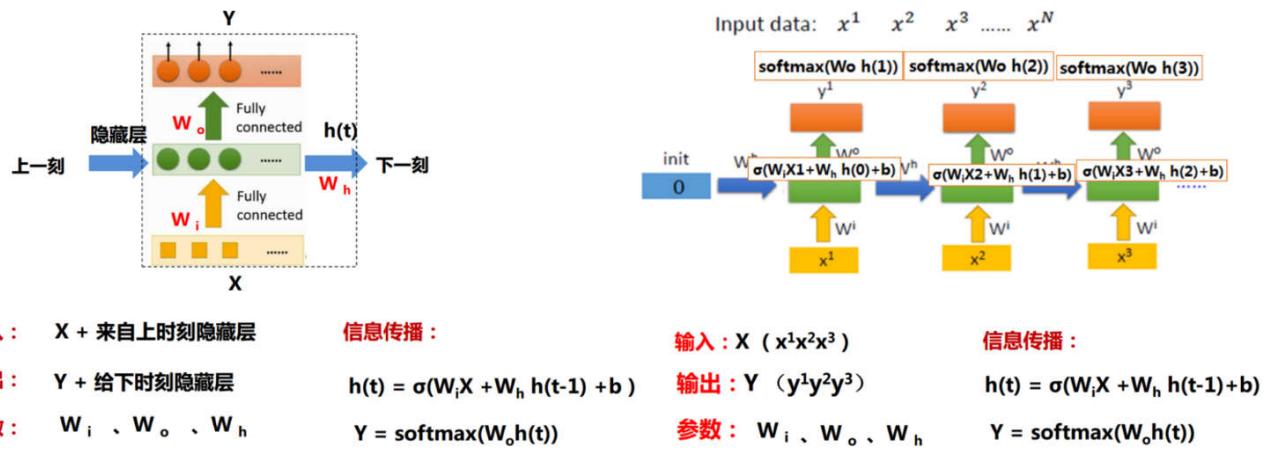


图2. RNN网络结构

可以看出RNN网络的输入输出均为序列数据，其网络结构本质上是一个小的全连接循环神经网络，只不过在训练时按时序张开。

RNN网络前向传播过程如下：

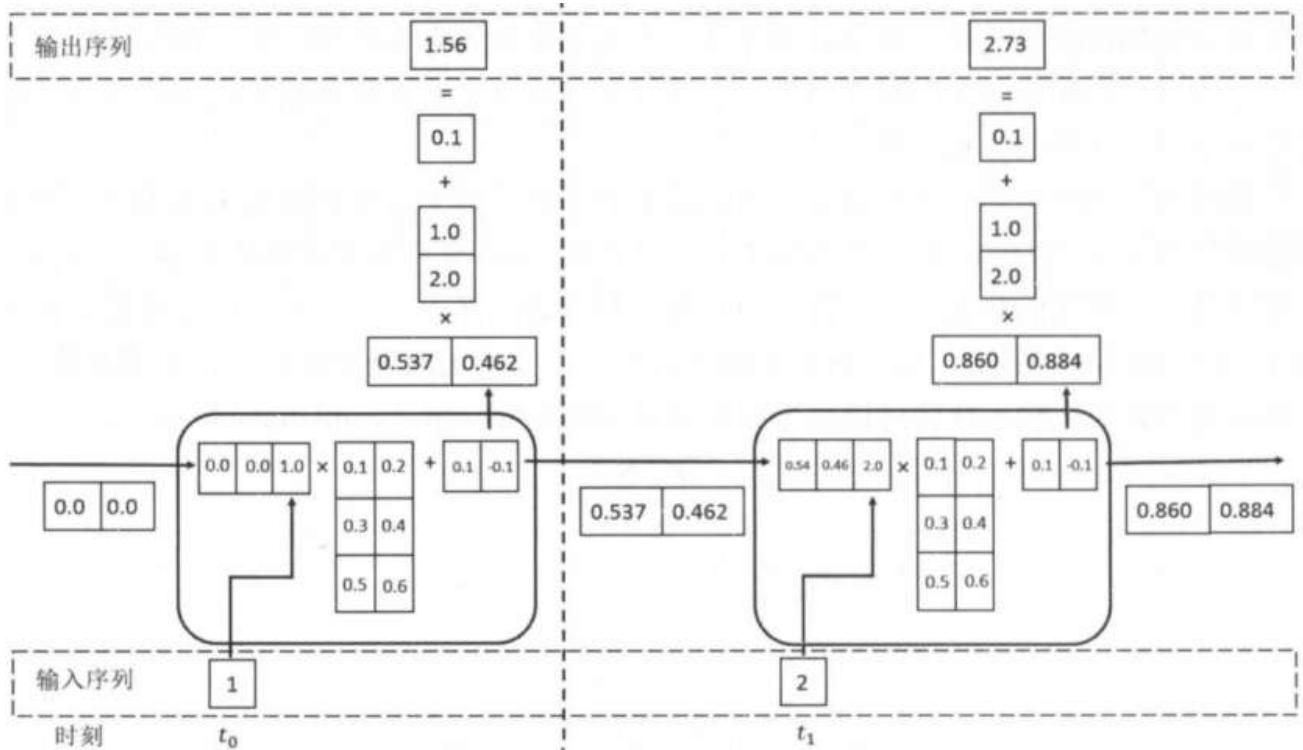
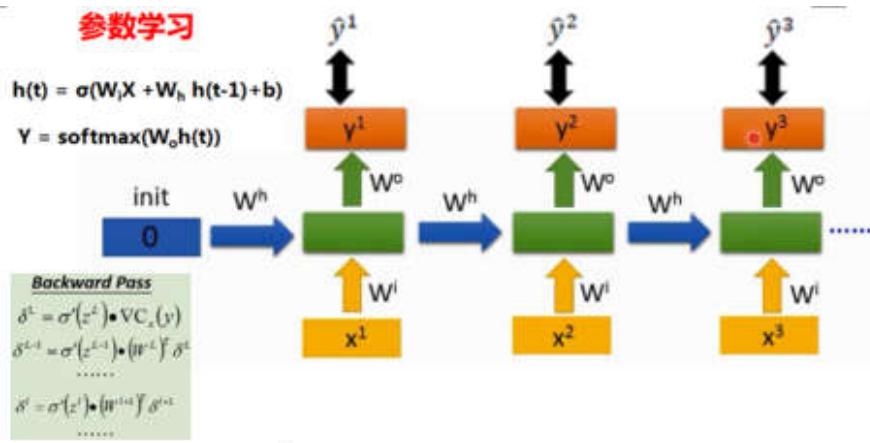


图3. RNN前向传播过程

RNN网络误差反向传播过程如下：



● 用 y^i 与 \hat{y}^i 的误差定义

损失函数： $L(\theta)$ 或 $C(\theta)$

$$\Theta = \{W_i, W_o, W_h, b\}$$

● 梯度下降法学习参数

$$w \leftarrow w - \eta \partial C^n / \partial w$$

$$W = (W^h)^T \text{diag}[f'(a_{i-1})]$$

参数共享 问题：

RNN 误差反传

1. 置同样初值

$$(W_{ij}^n)^0 = (W_{ij}^{n-1})^0$$

$$= (W_{ij}^{n-2})^0$$

$$\prod_{i=k+1}^n (W^h)^T \text{diag}[f'(a_{i-1})]$$

2. 同时调参数

$$[W_{ij}^n]^1 \leftarrow [W_{ij}^n]^0 - \eta \frac{\partial C(\theta)}{\partial W_{ij}^n} - \eta \frac{\partial C(\theta)}{\partial W_{ij}^{n-1}} - \eta \frac{\partial C(\theta)}{\partial W_{ij}^{n-2}}$$

$$[W_{ij}^{n-1}]^1 \leftarrow [W_{ij}^{n-1}]^0 - \eta \frac{\partial C(\theta)}{\partial W_{ij}^{n-1}} - \eta \frac{\partial C(\theta)}{\partial W_{ij}^n} - \eta \frac{\partial C(\theta)}{\partial W_{ij}^{n-2}}$$

$$[W_{ij}^{n-2}]^1 \leftarrow [W_{ij}^{n-2}]^0 - \eta \frac{\partial C(\theta)}{\partial W_{ij}^{n-2}} - \eta \frac{\partial C(\theta)}{\partial W_{ij}^n} - \eta \frac{\partial C(\theta)}{\partial W_{ij}^{n-1}}$$

the same memory

图4. RNN训练过程

RNN的训练过程与一般的DNN网络训练方法类似，采用BPTT的误差反向传播算法，在此不做过多的介绍。（有关BP算法的介绍具体可以参看我的这篇笔记）这里需要注意的是，RNN的训练过程是子网络沿时间轴的展开，其本质上仍在训练同一个子网络，因此在每次梯度下降时我们需要对同一个子网络的参数进行更新。常见的做法是在每一次参数的调整使均指向同一内存地址。

虽然RNN对于序列数据给出了较好的解决方案，然而对于较长的序列输入，为解决长期依赖问题，我们一般需要较深的神经网络，但是同一般的深度网络一样，RNN也存在优化困难的问题，即训练时间长，梯度消失 ($\prod_{i=k+1}^t (W^h)^T \text{diag}[f'(a_{i-1})] < 1$, 出现情况较少, 但对优化过程影响很大) 和梯度爆炸问题 ($\prod_{i=k+1}^t (W^h)^T \text{diag}[f'(a_{i-1})] > 1$, 大部分情况)。对于梯度消失问题，由于相互作用的梯度呈指数减少，因此长期依赖信号将会变得非常微弱，而容易受到短期信号波动的影响，对此可行的解决方法包括网络跨层连接、ESN、回声状态网络或引入leaky unit。而对于梯度爆炸问题，我们一般采取梯度截断的方法。目前解决RNN网络长期依赖问题更为普遍的做法是引入LSTM单元。

Rnn网络长期依赖问题：

- The cat, which already ate a bunch of food, (was) full.
- The cats, which already ate a bunch of food, (were) full.

长短时记忆神经网络 (LSTM)

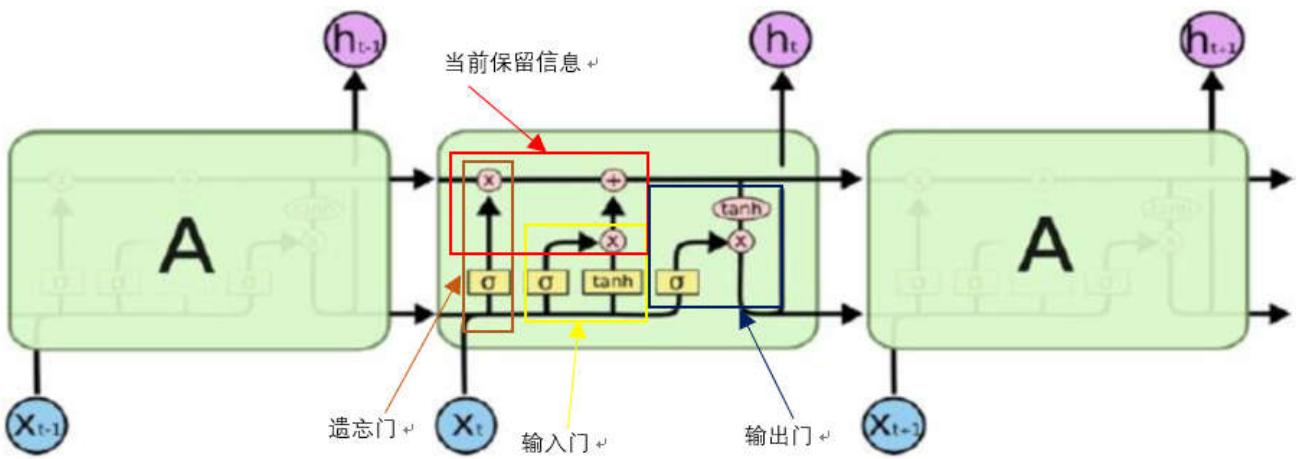
长短时记忆网络 (LSTM, long short-term memory) 结构是由Sepp Hochreiter和Jürgen Schmidhuber于1997年提出，所以其并非是一个深度学习的产物，在很多任务中采用LSTM结构的循环神经网络比标准的循环神经网络表现更好。LSTM与单一tanh循环体结构不同，其拥有三个特殊的“门”结构，它是一种特殊的循环体网络。

LSTM单元不仅接受此时刻的输入数据 x_t 和上一时刻的状态信息 h_{t-1} ，其还需建立一个机制能保留前面远处结点信息不会被丢失。具体操作是通过设计“门”结构实现保留信息和选择信息功能（遗忘门、输入门），每个门结构由一个sigmoid层和一个pointwise操作（按位乘法操作）构成。其中sigmoid作为激活函数的全连接神经网络层会输出一个0到1之间的数值，描述当前输入有多少信息量可以通过这个结构，其功能就类似于一扇门。

对于遗忘门，其作用是让循环神经网络“忘记”之前没有用的信息。遗忘门会根据当前的输入 x_t 和上一时刻输出 h_{t-1} 决定哪一部分记忆需要被遗忘。假设状态 c 的维度为 n ，“遗忘门”会根据当前的输入 x_t 和上一时刻输出 h_{t-1} 计算一个维度为 n 的向量 $f = \text{sigmoid}(W_1 x + W_2 h)$ ，其在每一维度上的值都被压缩在(0, 1)范围内。最后将上一时刻的状态 c_{t-1} 与 f 向量按位相乘，在 f 取值接近0的维度上的信息就会被“遗忘”，而 f 取值接近1的维度上的信息将会被保留。

在循环神经网络“忘记”了部分之前的状态后，它还需要从当前的输入补充最新的记忆。这个过程就是通过输入门完成的。输入门会根据 x_t 和 h_{t-1} 决定哪些信息加入到状态 c_{t-1} 中生成新的状态 c_t 。输入门和需要写入的新状态均是由 x_t 和 h_{t-1} 计算产生。

LSTM结构在计算得到新的状态 c_t 后需要产生当前时刻的输出，这个过程是通过输出门完成的。输出门将根据最新的状态 c_t 、上一时刻的输出 h_{t-1} 和当前的输入 x_t 来决定该时刻的输出 h_t 。通过遗忘门和输入门的操作循环神经网络LSTM可以更加有效地决定哪些序列信息应该被遗忘，而哪些序列信息需要长期保留，其结构如下图所示：



Neural Network Layer

Pointwise Operation

Vector Transfer

Concatenate

Copy

输入门 i_t

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

(决定加入多少新信息)

遗忘门 f_t :

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

(决定丢弃多少旧信息)

输出门 o_t

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

当前保留信息

隐状态输出

$$h_t = o_t * \tanh(C_t)$$

参数 : W_f, W_i, W_o, W_c

输入产生新信息:

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$

(在生成的当前保留信息中输入
产生新信息和旧信息各占多少)

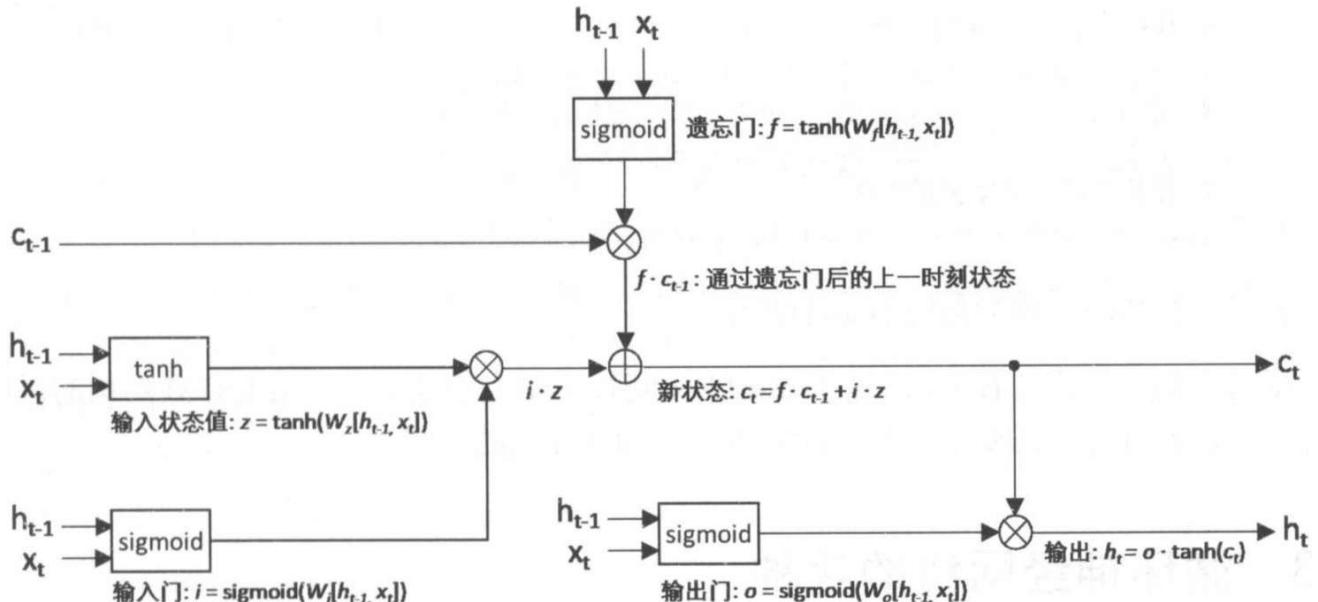


图5. LSTM单元结构

LSTM结构如上图所示，可以看出其结构较为复杂，这也引起了研究人员的思考：LSTM结构中到底那一部分是必须的？还可以设计哪些结构允许网络动态的控制时间尺度和不同单元的遗忘行为？对此Kyunghyun Cho et al.等人设计GRU单元(Gated recurrent unit)，以简化LSTM，如下所示：

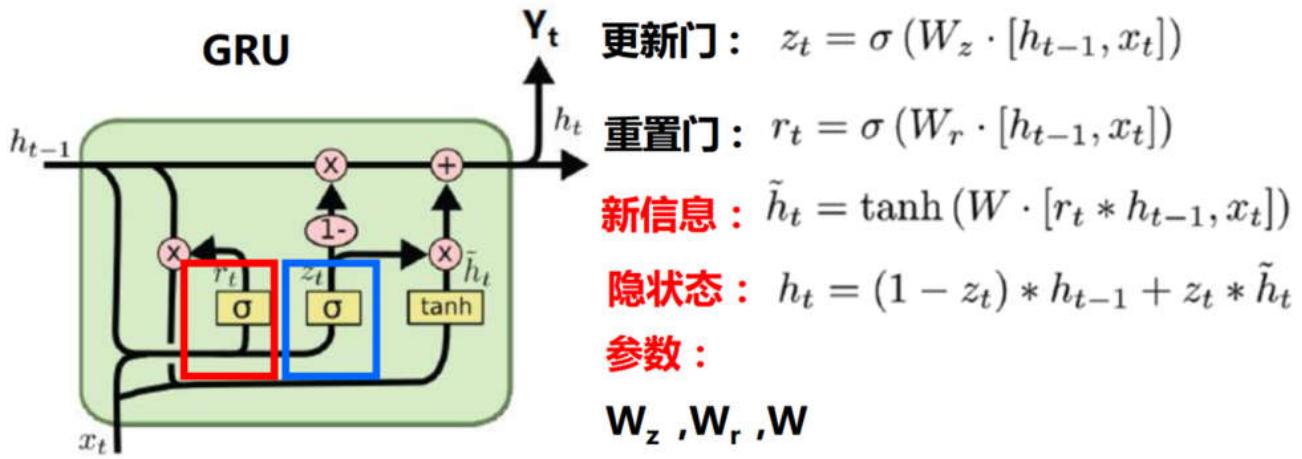


图6. GRU单元结构

如上图所示，GRU与LSTM最大的区别是其将输入门和遗忘门合并为更新门（更新门决定隐状态保留放弃部分）。然而在众多的LSTM变种中，其在很多任务中性能和鲁棒性均比不上GRU和LSTM。

RNN扩展结构

对于一般的RNN网络，其只有一个“因果”结构，即在时刻 t 的状态只能由过去的序列 x_1, x_2, \dots, x_{t-1} 以及当前的输入 x_t 确定。然而，在许多应用中，例如在语音识别中，由于协同发音和词与词之间的语义依赖，当前声音作为音素的正确解释可能取决于未来几个音素。即如果当前的词有两种或更多声学上合理的解释，我们可能要在更远的未来（和过去）寻找信息区分它们。因此输出的预测 y_t 可能依赖于整个输入序列。对此我们必须设计更新的网络结构同时结合上下文信息，双向RNN (Bidirectional RNNs) 应运而生。

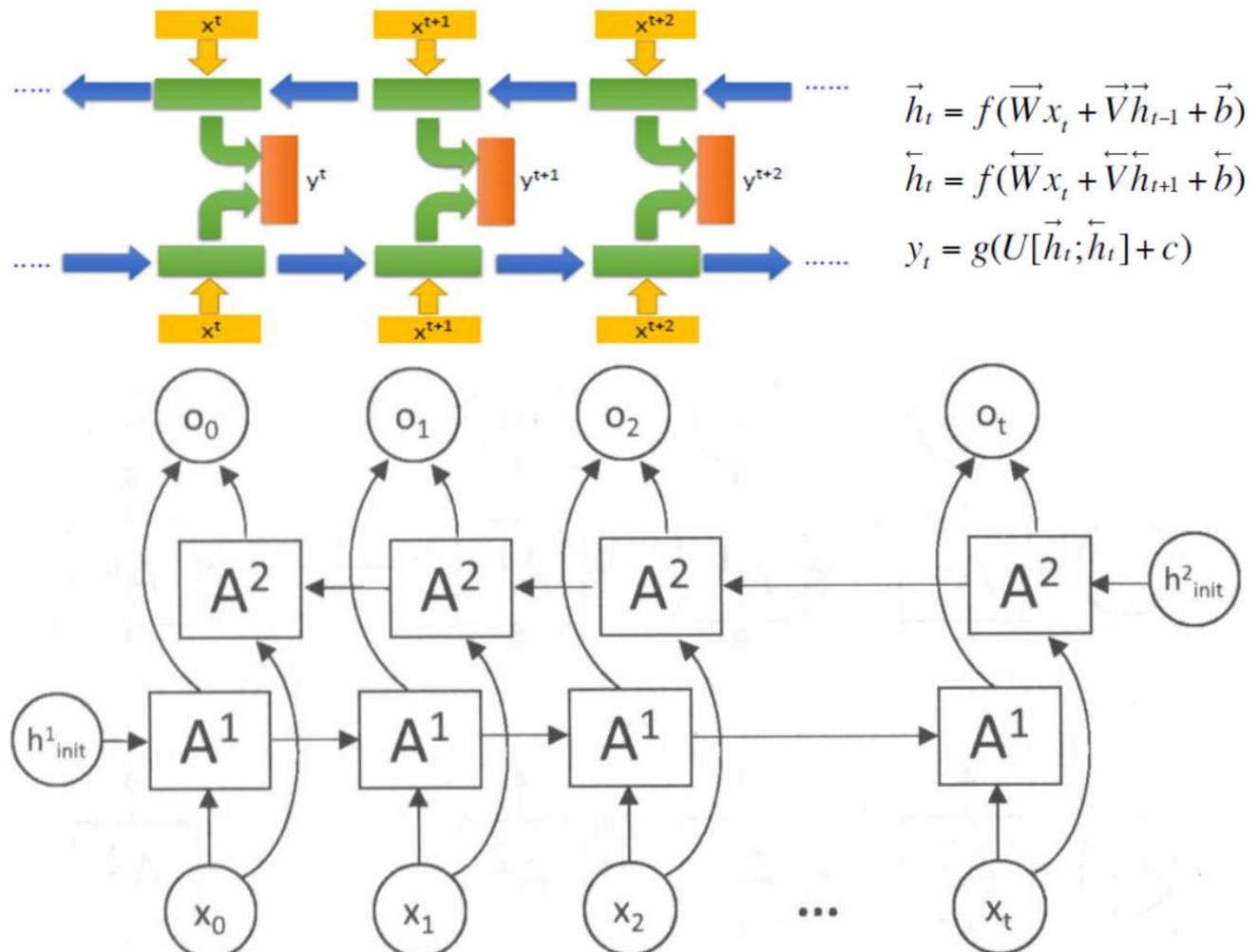


图7. 双向RNN网络结构

从图7中可以看出，双向循环神经网络的主体结构就是两个单向循环神经网络的结合。在每一个时刻，输入会同时提供给正向输入隐层（上文信息）和反向输入隐层（下文信息）。两个隐层（可看做两个独立的单向的RNN）独立进行计算，各自产生该时刻的新状态和输出，而双向循环网络的最终输出是这两个单向循环神经网络的输出的简单拼接。两个循环神经网络除方向不同以外，其余结构完全对称。每一层网络中的循环体可以自由选用任意结构，如简单的RNN、LSTM等。双向循环神经网络的前向传播过程和单向循环神经网络类似，这里不再赘述。更多关于双向神经网络的介绍可以参考Mike Schuster和Kuldip K. Paliwal发表的论文 Bidirectional recurrent neural networks。此外，双向RNN的思想也应用至二维输入，如图像中。对于图像其由4个RNN组成（上、下、左、右），如果RNN能够学习到承载长期信息，那在二维网格每个点的输出就能计算一个能捕捉到大多局部信息但仍依赖于长期输入的表示。相比卷积网络，应用于图像的RNN计算成本通常更高，但允许特征图间存在长期的相互作用。（Visin et al., 2015; Kalchbrenner et al., 2015）

另外还有Jordan Network、Deep RNN、Deep Bidirectional RNN等扩展，如下：

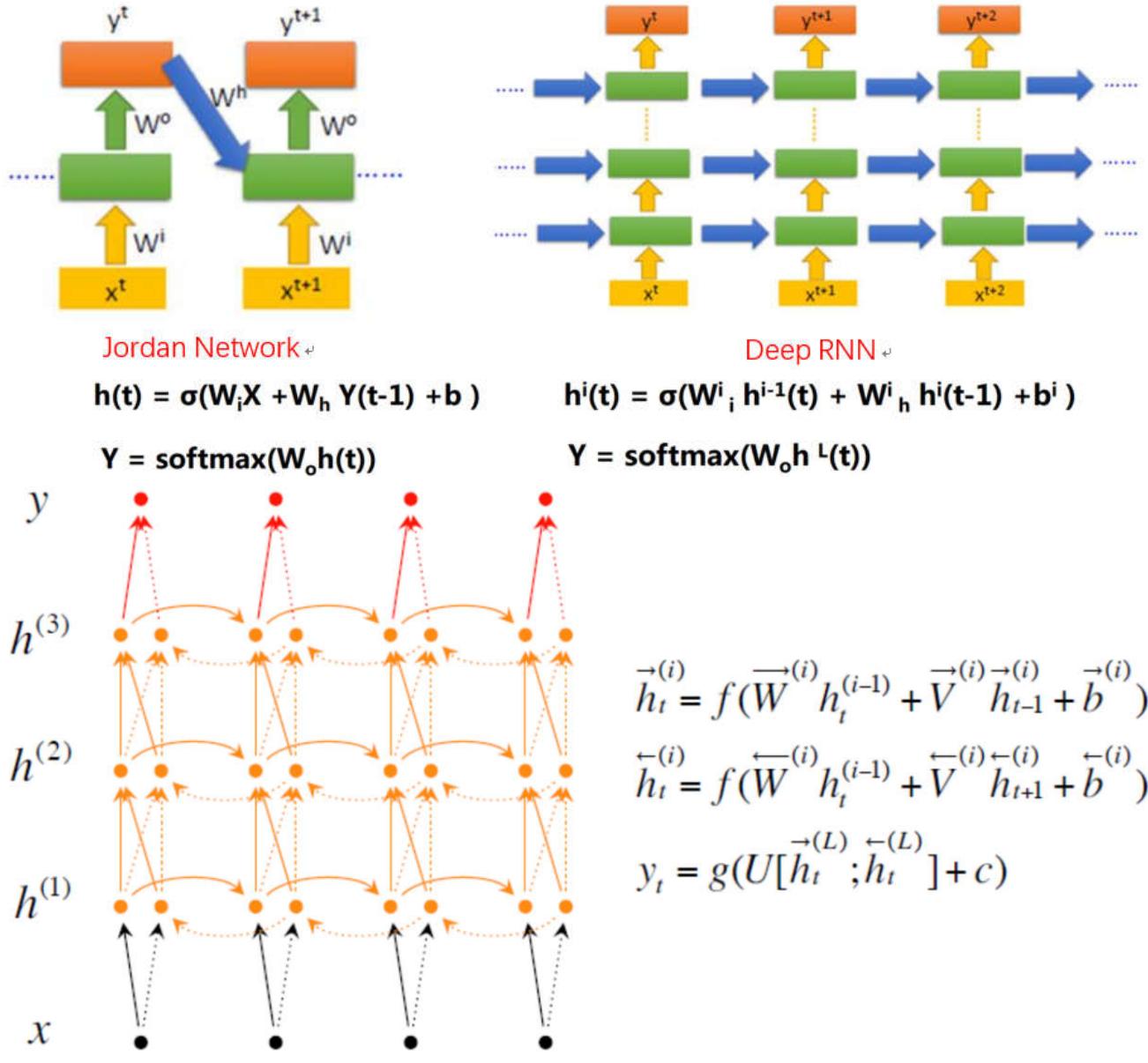


图8. Jordan Network、Deep RNN、Deep Bidirectional RNN网络结构

深层循环神经网络（Deep RNN）是循环神经网络的另外一种变种。为了增强模型的表达能力，可以在网络中设置多个循环层，将每层循环网络的输出传给下一层进行处理。对于普通的RNN，每一时刻的输入到输出之间只有一个全连接层，即在输入到输出的路径上是一个很浅的神经网络，因此从输入中提取抽象信息的能力将受到限制。而对于Deep RNN如图8所示，在每一时刻的输入到输出之间有多个循环体，网络因此可以从输入中抽取更加高层的信息。和卷积神经网络类似，每一层的循环体中参数是一致的，而不同层中的参数可以不同。

Encoder-Decoder框架

虽然LSTM确实能够解决序列的长期依赖问题，但是对于很长的序列（长度超过30），LSTM效果也难以让人满意，这时我们需要探索一种更有效的方法，即注意力机制（attention mechanism）。在介绍注意力机制前，我们先了解一种常用的框架：Encoder-Decoder框架。

在上文的讨论中，我们均考虑的是输入输出序列等长的问题，然而在实际中却大量存在输入输出序列长度不等的情况，如机器翻译、语音识别、问答系统等。这时我们便需要设计一种映射可变长序列至另一个可变长序列的RNN网络结构，Encoder-Decoder框架呼之欲出。

Encoder-Decoder框架是机器翻译（Machine Translation）模型的产物，其于2014年Cho et al.在Seq2Seq循环神经网络中首次提出。在统计翻译模型中，模型的训练步骤可以分为预处理、词对齐、短语对齐、抽取短语特征、训练语言模型、学习特征权重等诸多步骤。而Seq2Seq模型的基本思想非常简单——使用一个循环神经网络读取输入句子，将整个句子的信息压缩到一个固定维度（注意是固定维度，下文的注意力集中机制将在此做文章）的编码中；再使用另一个循环神经网络读取这个编码，将其“解压”为目标语言的一个句子。这两个循环神经网络分别称为编码器（Encoder）和解码器（Decoder），这就是encoder-decoder框架的由来。如下图所示：

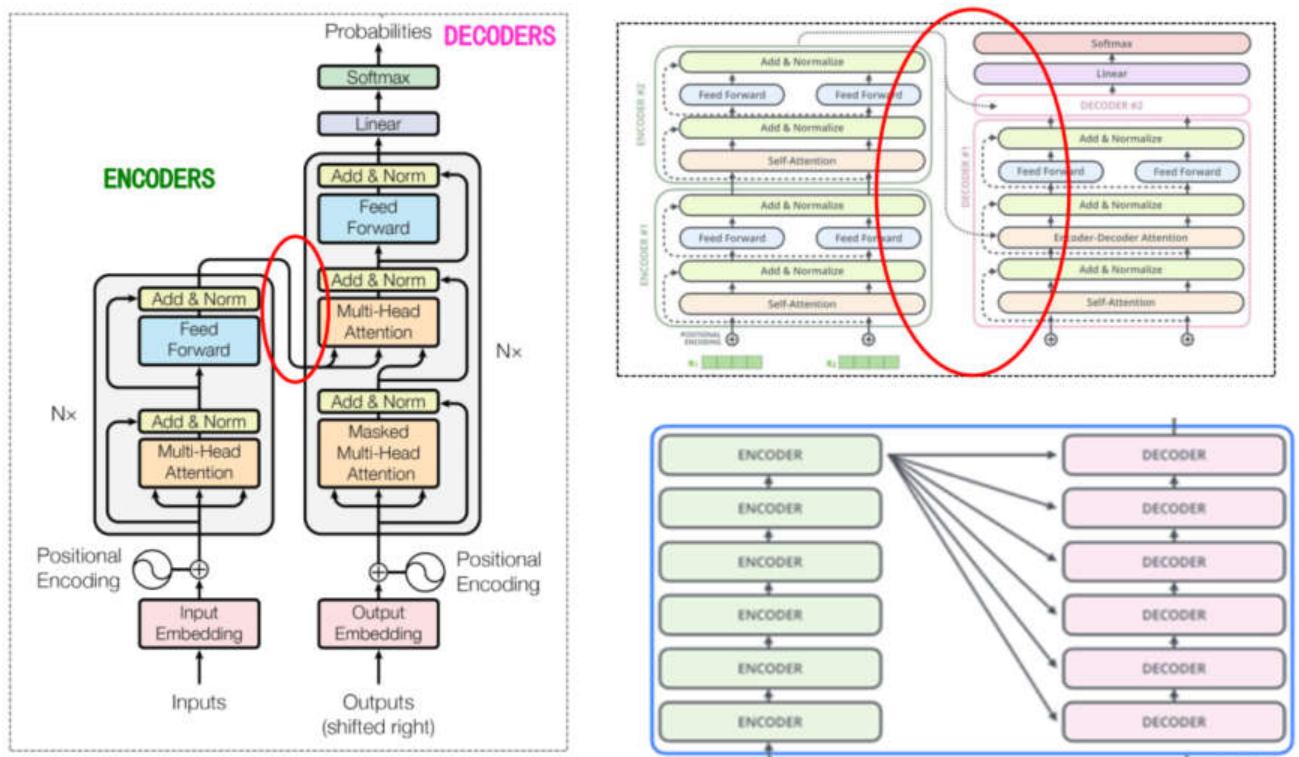


图9. Encoder-Decoder框架

Decoder: 根据 x 的中间语义表示 c 和已经生成的 y_1, y_2, \dots, y_{i-1} 来生成 i 时刻的 y_i , $y_i = g(c, y_1, y_2, \dots, y_{i-1})$ 。解码器部分的结构与一般的语言模型几乎完全相同：输入为单词的词向量，输出为softmax层产生的单词概率，损失函数为log perplexity。事实上，解码器可以理解为一个以输入编码为前提的语言模型。语言模型中使用的一些技巧，如共享softmax层和词向量的参数，均可以直接应用到Seq2Seq模型的解码器中。

Encoder: 对输入序列 x 进行编码，通过非线性变换转化为中间语义表示 c , $c = F(x_1, x_2, \dots, x_m)$ 。编码器部分网络结构则更为简单。它与解码器一样拥有词向量层和循环神经网络，但是由于在编码阶段并未输出最终结果，因此不需要softmax层。

Encoder-Decoder是一个十分通用的计算框架，其使用的具体模型如，CNN/RNN/Bi-RNN/GRU/LSTM/Deep LSTM等可根据不同的场景需求确定。此外，Encoder-Decoder框架其本质是实现直观

表示（例如词序列或图像）和语义表示之间来回映射。故通过该框架我们可以使用来自一种模态数据的编码器输出作为用于另一模态的解码器输入，以实现将一种模态转换到另一种模态的系统。正因为这个强大的功能，Encoder_Decoder框架以应用于机器翻译，图像生成标题等众多任务中。

注意力机制（Attention Mechanism）

Attention Mechanism最早引入至自然语言中是为解决机器翻译中随句子长度（超过50）增加其性能显著下降的问题，现已广泛应用于各类序列数据的处理中。机器翻译问题本质上可以看做是编码解码问题，即将句子编码为向量然后解码为翻译内容。早期的处理方法一般是将句子拆分为一些小的片段分别单独进行处理，深度学习通过构建一个大型的神经网络对同时考虑整个句子内容，然后给出翻译结果。其网络结构主要包括编码和解码两个部分，如双向RNN网络（Cho et al., 2014 and Sutskever et al., 2014）。编码器将一个句子编码为固定长度的向量，而解码器则负责进行解码操作。然而实验表明这种做法随着句子长度的增加，其性能将急剧恶化，这主要是因为用固定长度的向量去概括长句子的所有语义细节十分困难，这将需要足够大的RNN网络和足够长的训练时间。为克服这一问题，Neutral Machine Translation by Jointly Learning to Align and Translate文中提出每次读取整个句子或段落，通过自适应的选择编码向量的部分相关语义细节片段进行解码翻译的方案。即对于整个句子，每次选择相关语义信息最集中的部分同时考虑上下文信息和相关的目标词出现概率进行翻译。该文章最大的贡献就是首次提出了注意力集中机制（Attention Mechanism）（本质上是加权平均形成上下文向量），这在长句子的处理中得到了广泛的应用。

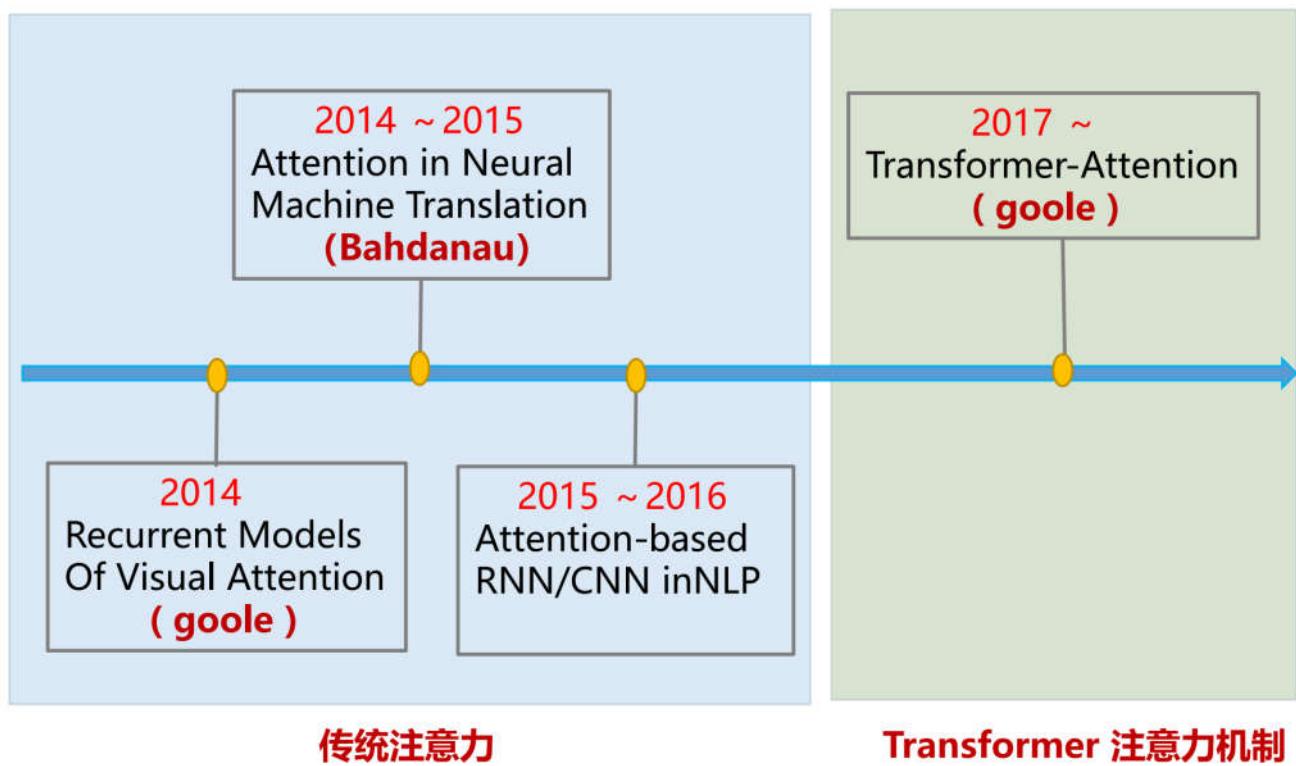


图10. Attention Mechanism演进历史

从上图可以看到注意力机制最早出现于图像标注领域，即对于同一场景，不同的人或任务会有不同的关注对象，如下：

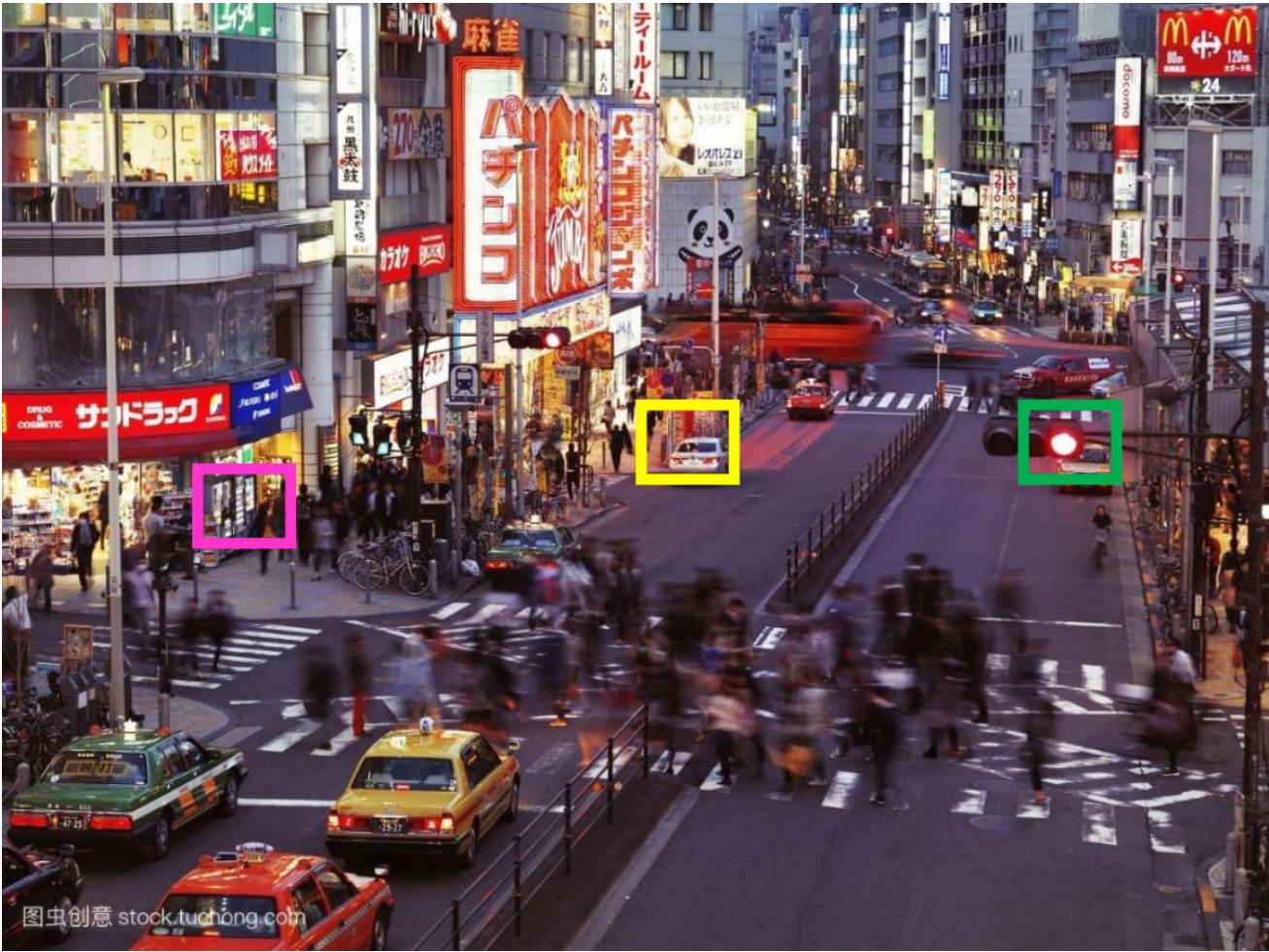


图11. Attention Mechanism在图像标注中的体现

注意力机制的主要亮点在于对于seq2seq模型中编码器将整个句子压缩为一个固定长度的向量 c ，而当句子较长时其很难保存足够的语义信息，而Attention允许解码器根据当前不同的翻译内容，查阅输入句子的部分不同的单词或片段，以提高每个词或者片段的翻译精确度。具体做法为解码器在每一步的解码过程中，将查询编码器的隐藏状态。对于整个输入序列计算每一位置（每一片段）与当前翻译内容的相关程度，即权重。再根据这个权重对各输入位置的隐藏状态进行加权平均得到“context”向量（Encoder-Decoder框架向量 c ），该结果包含了与当前翻译内容最相关的原文信息。同时在解码下一个单词时，将context作为额外信息输入至RNN中，这样网络可以时刻读取原文中最相关的信息，而不必完全依赖于上一时刻的隐藏状态。对比社seq2seq,Attention本质上是通过加权平均，计算可变的上下文向量 c 。

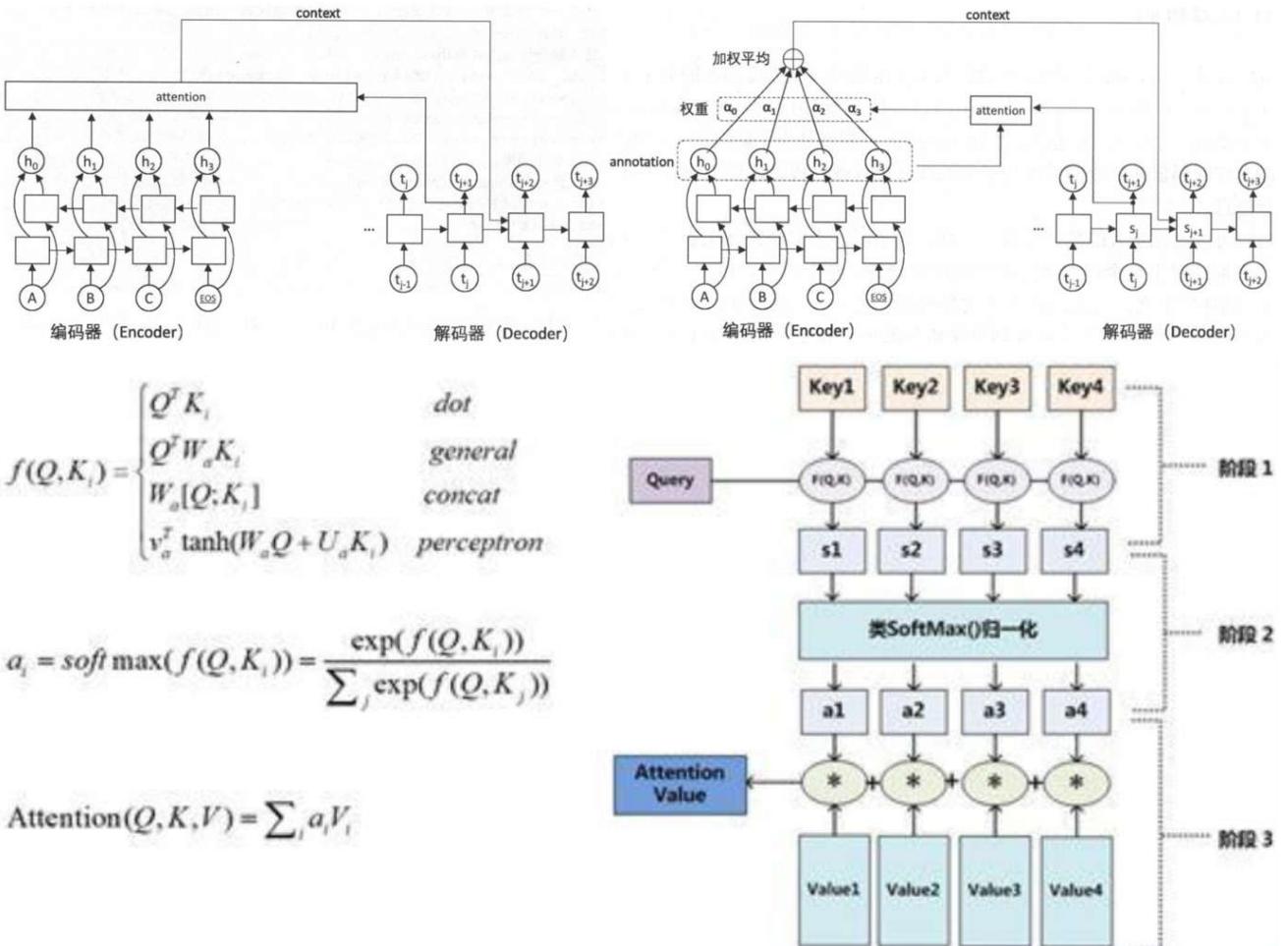
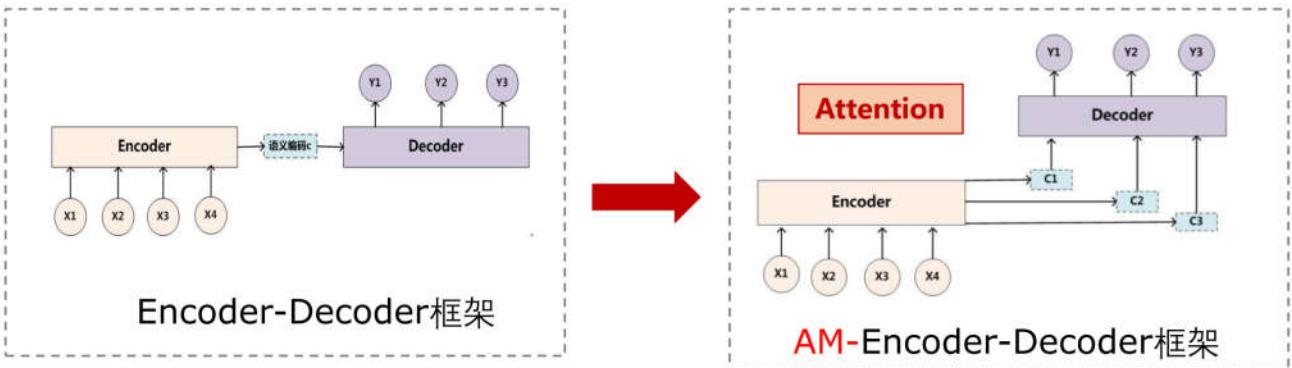


图12. Attention Mechanism

图12中的 K 为网络最终的输出结果； Q 即为Attention-Value； $f(Q, K)$ 为注意力打分函数（包括多种形式，如乘法公式：dot、general、concat；加法公式：perceptron），其值 α 的大小反应输出对输入不同片段的重视程度； V 即为输入词向量；其最终的输出结果 $\text{Attention}(Q, K, V)$ 即为context向量，其包含了此时输出片段对输入所关注内容的全部信息。在GNMT中其context的计算如下所示：

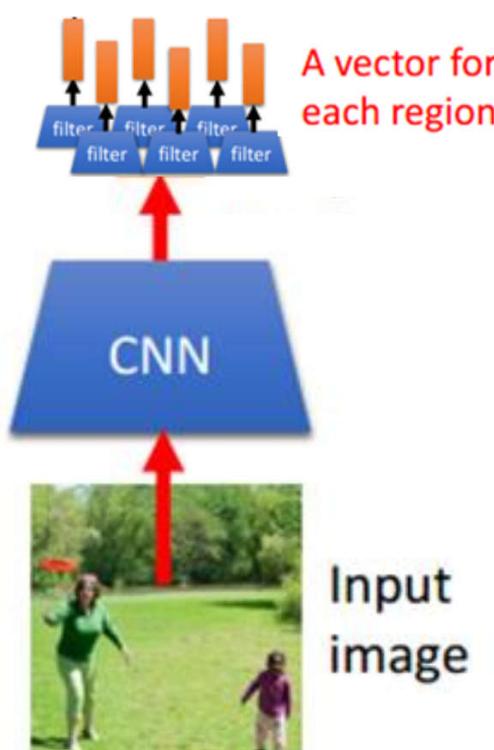
其中，第 j 时刻的 context_j 计算如下：

$$\begin{aligned} \alpha_{ij} &= \frac{\exp(e(h_i, s_j))}{\sum_i \exp(e(h_i, s_j))} \\ e(h, s) &= Utanh(Vh + Ws) \\ \text{context}_j &= \sum_i \alpha_{i,j} h_i \end{aligned} \quad (1)$$

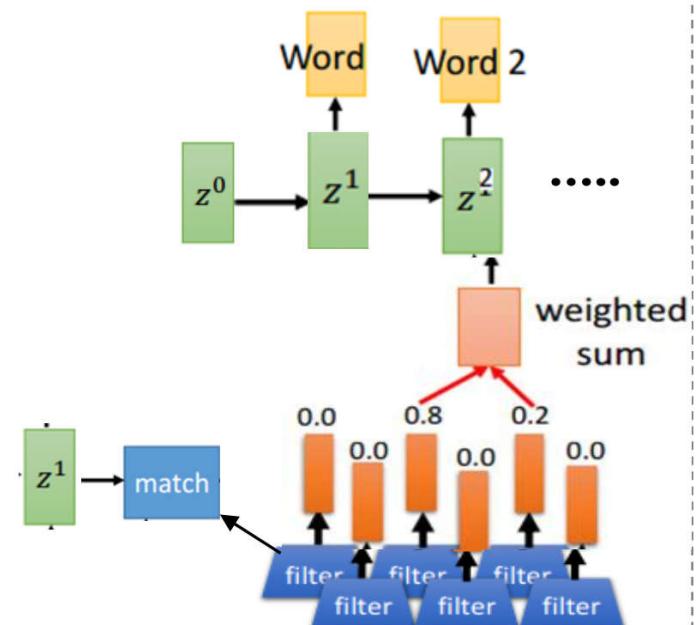
上式中， U, V, W 为模型参数。 h_i 表示编码器在第 i 个单词上的输出， s_j 为编码器预测第 j 个单词的状态， α 为通过Softmax计算的权值， $e(h, s)$ 为计算原文各单词与当前解码器状态的“相关度”函数，其构成了包含一个隐藏层的全连接神经网络。

除上式外，注意力机制还有多种其他设计，如Minh-Thang Luong等人提出的 $e(h, s) = h^T W s$ 。或直接使用两个状态之间的点乘 $e(h, s) = h^T s$ 。需要注意的是无论采用哪个模型的“相关度”函数 $e(h, s)$ ，通过softmax计算权重 α 和通过加权平均计算context的方法都是一样的。对比Attention和seq2seq可以发现主要有两点差别：（1）Attention编码器采用了一个双向循环网络。虽然seq2seq模型也可以使用双向循环网络作为编码器，但是在注意力机制中，这一设计必不可少。其主要是因为解码器通过注意力查询一个单词时，通常需要知道该单词周围的部分信息，而双向RNN通常能实现这一要求。（2）Attention中取消了编码器和解码器之间的连接，解码器完全依赖于注意力机制获取原文信息。取消这一连接使得编码器和解码器可以自由选择模型。例如它们可以选择不同层数、不同维度、不同结构的循环神经网络，可以在编码器中使用双向LSTM，而在解码器使用单向LSTM，甚至可以用卷积网络作为编码器、用循环神经网络作为解码器等。

- Input an image, but output a



RNN



A woman is throwing a Frisbee in a park

- Good captions



A woman is throwing a frisbee in a park.



A dog is standing on a hardwood floor.



A stop sign is on a road with a mountain in the background.



A little girl sitting on a bed with a teddy bear.



A group of people sitting on a boat in the water.



A giraffe standing in a forest with trees in the background.

图13. 图片标题的生成

上图为Attention Mechanism在图像标题生成中的应用，从中可以明显看出标题中的描述内容（下划线部分）在图片中即对应高亮区域。此外attention mechanism的一些变式，如Hard AM、Soft AM、Global AM、Local AM的结构如下：

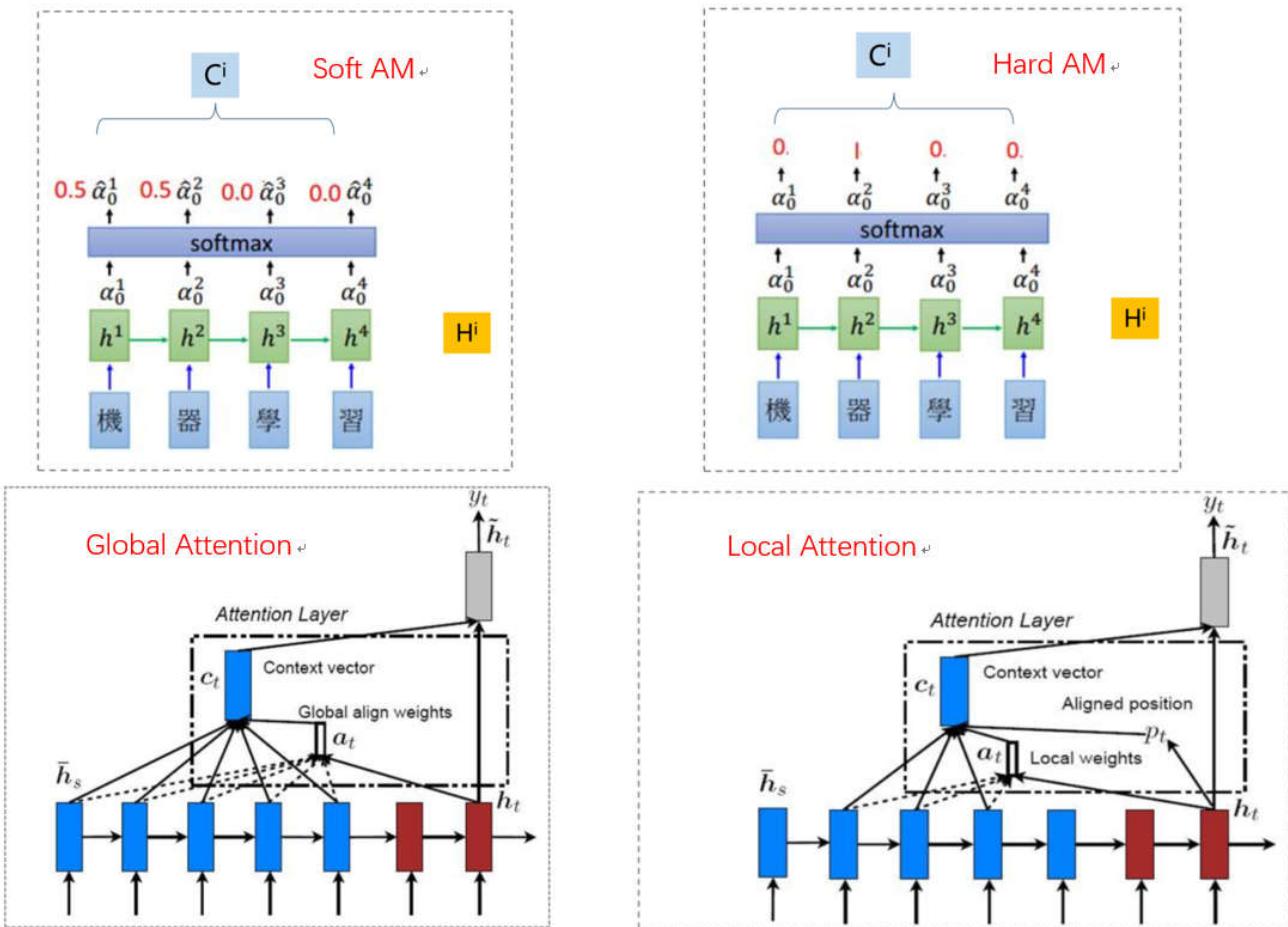


图14. Attention Mechanism的变种

- Soft AM: 在注意力分配概率分布的计算过程中, 对于输入句子 X 中任意一个单词都给出对应的概率, 为概率分布;
- Hard AM: 直接从输入句子里面找到某个特定的单词, 然后将目标句子单词和这个单词对齐, 而其它输入句子中的单词规定对齐概率为0;
- Global AM: Decode端Attention的计算考虑输Ecoder端序列中所有词汇;
- Local AM: Local Attention Model本质上是Soft AM和 Hard AM的一个混合或折中。即一般首先预估一个对齐位置 P_t , 然后在 P_t 左右大小为D的窗口范围内计算类似于Soft AM的概率分布。

注意力机制是一种高效获取信息的方式。一方面, 它使得解码器可以在每一步主动查询最相关的信息, 而暂时忽略不相关的信息; 另一方面, 它大大缩短了信息流动的距离。在传统的seq2seq模型中, 如果解码器生成最后一个单词时需要用到编码器读入的第一个单词的信息, 那么这个信息需要通过所有的LSTM节点才能从编码器的最前端传递到编码器的最后端, 而有了注意力机制后, 解码器在任意时刻只需一步就可以查阅输入的任意单词。鉴于这些优点, 注意力机制在很多模型中得到了广泛应用。例如ConvSeq2Seq模型使用卷积神经网络取代了Seq2Seq模型的循环神经网络, 同时仍使用相似的注意力机制在编码器和解码器之间传递信息。Transformer模型既不使用循环神经网络, 也不使用卷积神经网络, 而完全使用注意力机制在不同神经层之间传递信息, 并在机器翻译任务中取得了不错的效果。在图像领域中, 注意力机制和卷积神经网络结合, 在图像分类、图片描述生成等应用上取得了很好的效果。

Machine Translation

机器翻译问题最早是由W.Weaver于1949年正式提出, 直至今日已过去了大半个世纪。而使用的策略也从早期主要基于规则发展为如今深度学习的方法, 机器翻译问题是在NLP中是一个环境相对简单纯粹的问题, 因此在自然语言发展史上每一次技术的突破都伴随有机器翻译质量的提高。机器翻译主要的方法包括以下几种: 统计机器翻译 (Statistical Machine Translation, SMT), 基于规则的机器翻译 (Rule-Based Machine Translation, RBMT) (语言知识由语言专家人工编写), 基于实例的机器翻译方法 (语言知识来源语料

库），翻译记忆方法（基于实例翻译法的一种特例），以及其方法的结合（Hybrid Systems）。无论是人或机器，翻译结果都要求尽可能接近原文所表达的意思，减少失真，因此翻译绝不是每个词汇的简单替代。对于人工翻译，我们必须分析句法、语法、句子结构结合上下文甚至句子内容所涉及的领域知识去理解句子含义。早期，众多的方法均将关注点放在individual language pairs的构建上，因为构建一个单一的系统去完成多种语言间的相互翻译实在是太困难了。2014年Sutskever等人提出seq2seq模型（Encoder-Decoder框架）首次实现了End-to-End的机器翻译。2015年，Bahdanau等人发明了注意力机制缓解了长句子性能急剧下降的问题，并将其简单的应用于多种语言的翻译任务中。2015年，Dong等人通过增加一系列的decoder和attention mechanism以实现multilingual machine translation。同样Luong等人也在多语言翻译上进行了尝试，与Dong不同的是其只使用一个attention mechanism而扩展了多个encoders and decoders。2016年Caglayan et al., Zoph and Knight, Firat et al., Lee et al.等多位学者也分别针对multilingual machine translation提出了不同的方案。然而这些方法要么需要针对不同的语言设计不同的encoder-decoder or attention mechanism，要么需要multi-way parallel corpus，要么只能实现针对一种target language的translation。

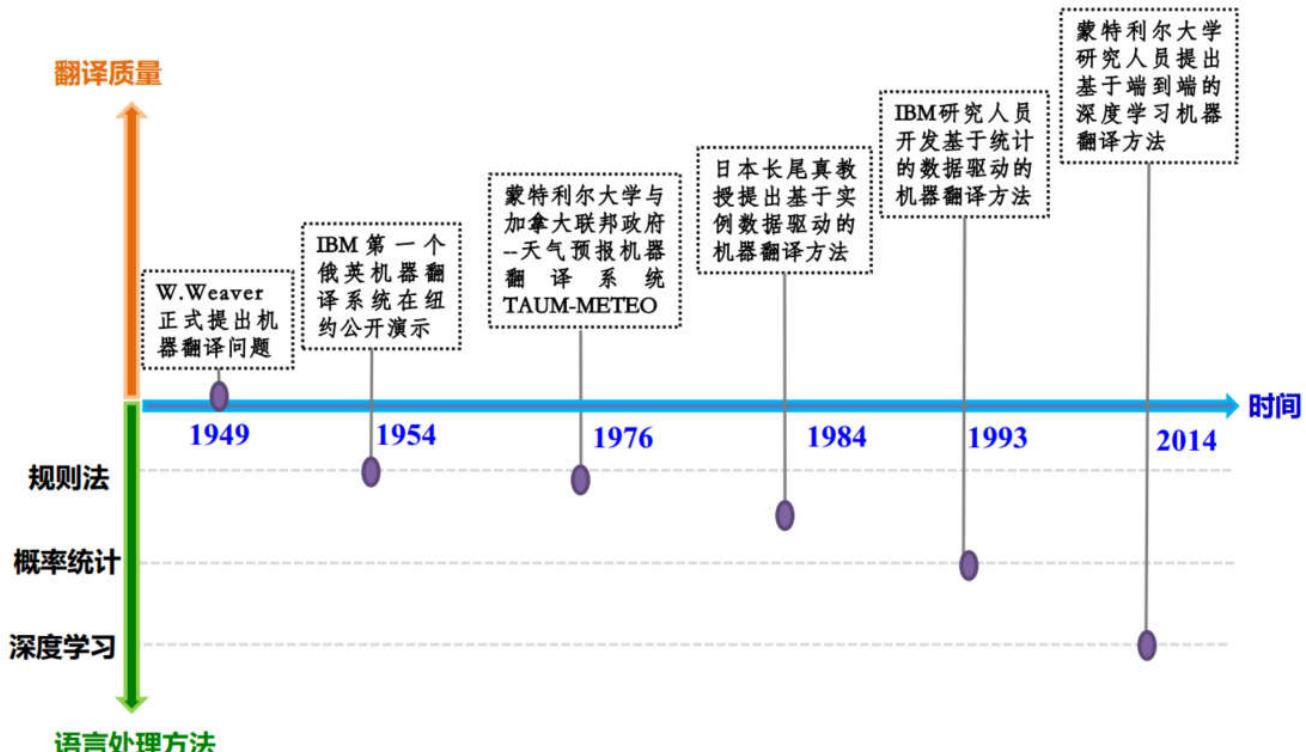


图15. 机器翻译的发展历史

早期的机器翻译方法：

(1) 基于规则机器翻译方法

规则机器翻译系统基本步骤主要包括：

- Step1. 字典的构建；
- Step2. 词语切分；
- Step3. 引入人名识别与翻译；
- Step4. 引入词性标注；
- Step5. 引入句法分析；
- Step6. 引入翻译规则：插入译文词、调整语序、多层次调序与插入；
- Step7. 引入目标语言属性。

基于规则方法的优点主要有：

- 直观，能够直接表达语言学家的知识；

- 规则的颗粒度具有很大的可伸缩性。大颗粒度的规则具有很强的概括能力，小颗粒度的规则具有精细的描述能力；
- 便于处理复杂的结构和进行深层次的理解，如解决长距离依赖问题；
- 大颗粒度的规则具有较强的系统适应性，不依赖于具体的训练语料。

基于规则方法的缺点主要有：

- 规则的覆盖性差，特别是细颗粒度的规则很难总结得比较全面。较强的规则其覆盖性低，但较为准确（准确率高而召回率低），较弱的规则覆盖全面，但会包含许多“垃圾”（准确率低而召回率高）（跷跷板现象）；
- 规则之间的冲突没有好的解决办法（跷跷板现象）；
- 规则一般只局限于某一个具体的系统，迁移性差（这也是自然语言处理中各个任务的一个难点），规则库开发成本太高。

(2) 基于实例的机器翻译方法

基于实例的机器翻译方法主要由长尾真(Makoto Nagao)在1984年发表的《采用类比原则进行日-英机器翻译的一个框架》中提出，其主要思想为：人类并不通过做深层的语言学分析来进行翻译，人类的翻译过程是：首先把输入的句子正确地分解为一些短语碎片，接着把这些短语碎片翻译成其它语言的短语碎片，最后再把这些短语碎片构成完整的句子，每个短语碎片的翻译是通过类比的原则来实现的。因此，应该在计算机中存储一些实例，并建立由给定的句子找寻类似例句的机制，这是一种由实例引导推理的机器翻译方法，也就是基于实例的机器翻译。在基于实例的机器翻译系统中，翻译知识以实例和义类词典的形式来表示，系统的主要是双语对照的翻译实例库，实例库主要有两个字段，一个字段保存源语言句子，另一个字段保存与之对应的译文，每输入一个源语言的句子时，系统把这个句子同实例库中的源语言句子字段进行比较，找出与这个句子最为相似的句子，并模拟与这个句子相对应的译文，最后输出译文。如果利用了较大的翻译实例库并进行精确的对比，就有可能产生高质量译文，而且避免了基于规则的那些传统的机器翻译方法必须进行深层语言学分析的难点。此种翻译策略在早期的实际应用中取得了不错的效果，得到了广泛的应用。

基于实例方法的优点包括：

- 使用语料库作为翻译知识来源，无需人工编写规则系统开发成本低，速度快；
- 从语料库中学习到的知识比较客观；
- 从语料库中学习到的知识覆盖性比较好；

然而其缺点也十分明显：

- 系统性能依赖于语料库，质量较高的语料库难以获得且开发成本高，而专有领域则更是如此；
- 数据稀疏问题严重
- 语料库中不容易获得大颗粒度的高概括性知识。

(3) 翻译记忆方法

翻译记忆方法(Translation Memory)是基于实例方法的特例，也可以把基于实例的方法理解为广义的翻译记忆方法。其主要思想包括以下两步：(1) 把已经翻译过的句子保存起来；(2) 翻译新句子时，直接到语料库中去查找。如果发现相同的句子，直接输出译文，否则交给别人去翻译，但可以提供相似的句子的参考译文。

基于统计的机器翻译方法：

1946年美国洛克菲勒基金会(Rockefeller Foundation)副总裁W.Weaver提出机器翻译的想法；1949年，韦弗发表了一份以《翻译》为题的备忘录，提出：“当我阅读一篇用汉语写的文章的时候，我可以说，这篇文章实际上是用英语写的，只不过它是用另外一种奇怪的符号编了码而已，在阅读时，我是在进行解码”。韦弗的卓越思想成为了而后统计机器翻译(Statistic Machine Translation,简称SMT)的理论基础。1990年IBM的

Peter F. Brown等人在Computational Linguistics上发表论文“统计机器翻译方法”。1993年又再次在该杂志上发表论文“统计机器翻译的数学：参数估计”，这两篇文章奠定了统计机器翻译的理论基础。统计机器翻译方法主要包括：1.基于词的翻译方法；2.基于短语的翻译方法；3.基于句法的翻译方法，分为：a.基于层次化短语方法（形式语法）；b.基于树的方法（语义句法）。

(1) 基于词的翻译方法

基于词的翻译方法代表模型有IBM统计翻译模型 (IBM 1-5)，即噪声信道模型。其主要思想为：一种语言 T 由于经过一个噪声信道而发生变形，从而在信道的另一端呈现为另一种语言 S （信道意义上的输出，翻译意义上的源语言）。翻译问题实际上就是如何根据观察到的 S ，恢复最为可能的 T 问题。这种观点认为，任何一种语言的任何一个句子都有可能是另外一种语言中的某个句子的译文，只是其可能性有大有小，[Brown et. al, 1990]。故基于词的翻译问题即可抽象、转化为如下问题：

给定源语言句子 $S = s_1^m = s_1s_2\dots s_m$ 及目标句子 $T = t_1^l = t_1t_2\dots t_l$ （输入），其目标为最大化 $t, t^* = \text{argmax}_t P(t|s)$ （输出）。

这里我们使用贝叶斯求解：

$$P(T|S) = \frac{P(T)P(S|T)}{P(s)}$$

$$T' = \text{argmax}_T P(T)P(S|T)$$

上式中的 $P(T)$ 即为语言模型， $P(S|T)$ 即为翻译模型。

故以上问题又可转化为如下三个问题：

- 估计语言模型概率 $P(T)$

目标语言模型学习问题，可通过各类语言模型解决。

- 估计翻译概率 $P(S|T)$

翻译时词之间的对应情况，不同定义的词对齐关系即有不同的翻译模型。

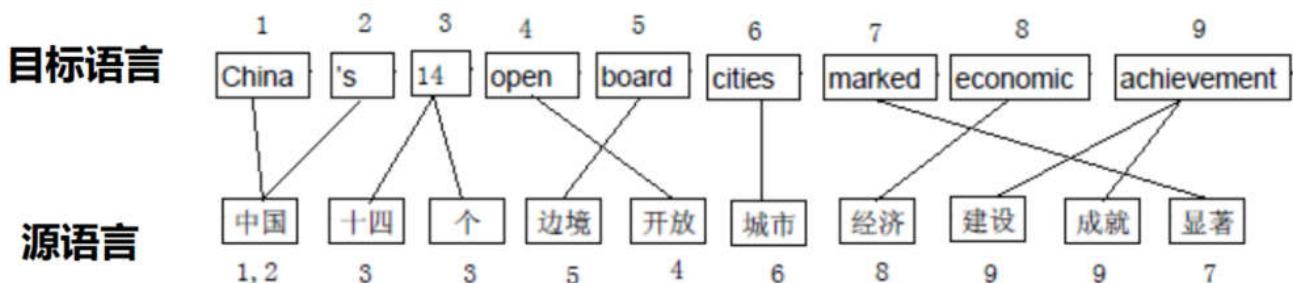


图16. 词对齐

- 快速有效地搜索 T ，使得 $P(T)P(S|T)$ 最大

给定 S ，求 T ，使得 $P(T)P(S|T)$ 最大，采用的方法主要包括：贪心算法、堆栈搜索等。

虽然基于词的统计机器翻译方法其性能已被后来的各种方法所超越，但是其词对齐的思想却成为了统计机器翻译方法的基础（Encoder-Decoder将词对齐也颠覆了）。

由于基于词的翻译模型只刻画了词到词的翻译概率，词翻译的时候没有考虑上下文，故在词语调序方面能力很差。其难以刻画一些固定搭配、习惯用法的翻译，很难处理词义消歧问题，很难处理一对多、多对一和多对多的翻译问题。故很多研究者通过在短语层面进行建模，以改进局部词语调序的效果。主要包括Och、Zens、Koehn 等人的工作。（NLP的基本核心任务本质上是消歧问题，无论是分词，序列标注，还是语义分

析等等。而消歧问题本质上又是分类问题，有无歧义对应0-1分类，具体是哪一种解对应多分类即多选一问题，因此这也是机器学习的方法（传统又或深度）能够得到广泛应用的原因）

(2) 基于短语的翻译方法

基于短语的翻译方法代表模型如Koehn，2003年提出基于短语的对数线性模型翻译模型，其基本思想为以短语为基本翻译单元 把训练语料库中所有对齐的短语及其翻译概率存储起来，作为一部带概率的短语词典，翻译的时候将输入的句子与短语词典进行匹配，选择最好的短语划分，按短语进行翻译，然后将得到的短语译文重新排序，得到最优的译文。短语通常是指一个连续的词串(n-gram)，其不一定是语言学中定义的短语(phrase)。如：我想预订一个单人间 —— I would like to reserve a single room。其中“我想 —— I would like to”即为一个短语。

基于词翻译：



基于短语翻译：

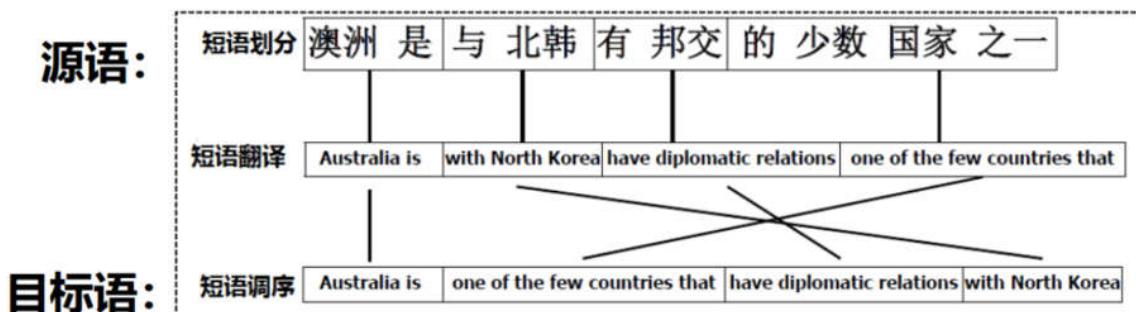
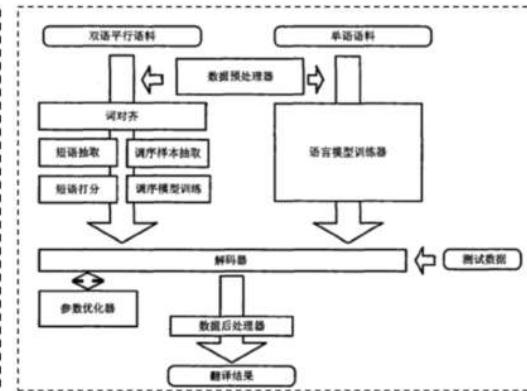


图17. 基于词翻译与基于短语翻译方法对比

基于短语的统计翻译过程



基于短语的对数线性模型翻译系统



$$\text{线性对数模型} \quad T' = \operatorname{argmax}_T P(T | S) = \operatorname{argmax}_T \frac{\exp\{\sum_1^M \lambda_m h_m(T, S)\}}{\sum_{T^*} \exp\{\sum_1^M \lambda_m h_m(T^*, S)\}}$$

$$= \operatorname{argmax}_T \left\{ \sum_1^M \lambda_m h_m(T, S) \right\}$$

图18. 基于短语的翻译模型

(3) 基于句法的翻译方法

基于句法的翻译方法主要包括：(a) 基于层次化短语方法（形式句法），如括号转录语法；(b) 基于树的方法（语义句法），如同步树替换文法(STSG)。当使用句法分析方法进行机器翻译，需要双语同步上下文无关文法(SCFG)。

(a) 基于层次化短语的翻译方法

基于层次化短语方法，代表模型包括基于层次短语的翻译模型[David Chiang,2005]，其核心是引入了嵌套层次短语的思想，并采用括号转录语法（同步上下文无关语法的特例）作为形式化方法，其所有句法结构规则（短语模板）可不使用任何语言学知识直接从平行语料库中自动学习得到，在完成源语言句法分析的同时，目标语言随即生成，因此可利用各种成熟的句法分析算法进行机器翻译，而无需另外设计专门的翻译算法。该方法易于实现，解码过程复杂度相对较低，效果比传统短语模型有很大提高。

如，翻译过程

文法规则：

- (1) $X \rightarrow <与 X_1 \text{ 有 } X_2, \text{ have } X_2 \text{ with } X_1>$
- (2) $X \rightarrow <X_1 \text{ 的 } X_2, \text{ the } X_2 \text{ that } X_1>$
- (3) $X \rightarrow <X_1 \text{ 之一, one of } X_1>$
- (4) $X \rightarrow <\text{澳洲, Australia}>$
- (5) $X \rightarrow <\text{邦交, diplomatic relations}>$
- (6) $X \rightarrow <\text{少数国家, few countries}>$
- (7) $X \rightarrow <\text{北韩, North Korea}>$
- (8) $(S \rightarrow S_1 X_2, S_1 X_2)$
- (9) $(S \rightarrow X_1, X_1)$

源语句子

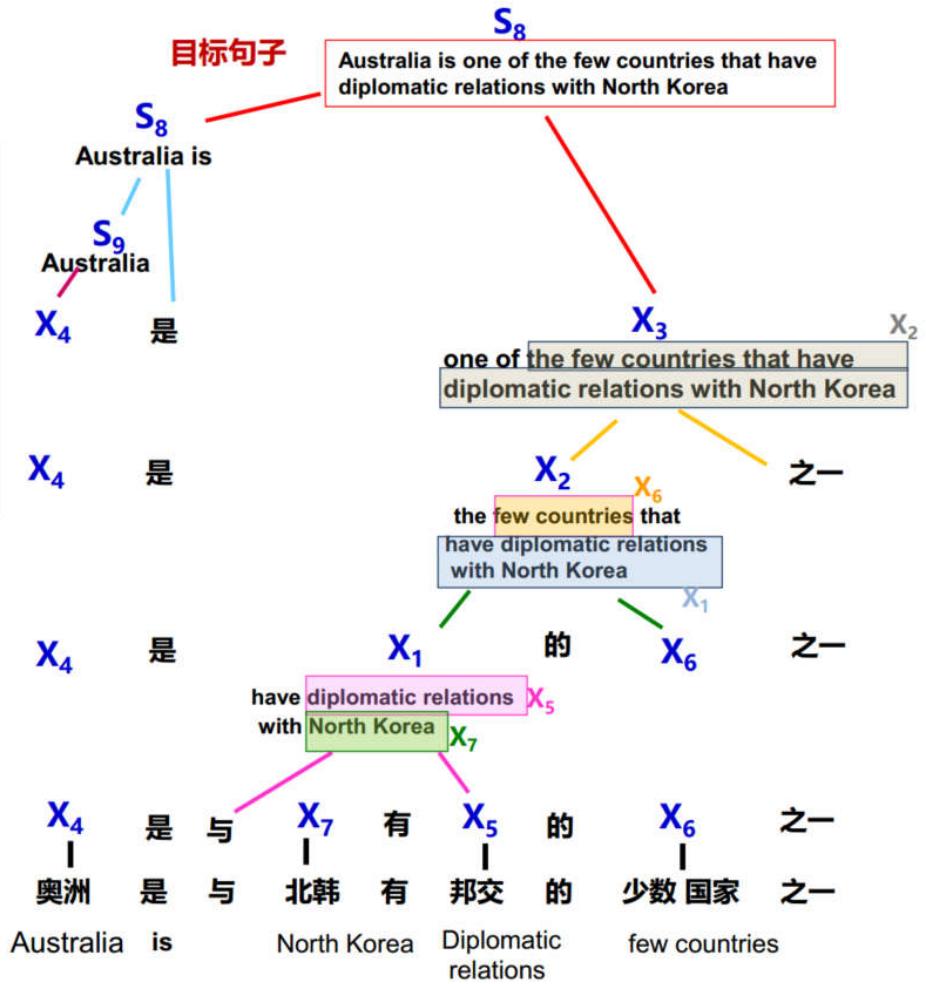


图19. 基于层次化短语的翻译模型

(b) 基于树的翻译方法

基于树的翻译模型其特点主要包括：树翻译模型属于语言学上基于句法的模型使用语言学知识，语言知识规则通常从句法树库训练得到。翻译模型通常依据句法分析方法进行机器翻译。该模型可以分为：树到树模型：在源语言端和目标语言端都使用语言知识；树到串模型：只在源语言端使用语言知识；串到树模型：只在目标语言端使用语言知识。

- 树到树的代表模型，如Zhang et al.(2007, 2008)。其建模过程包括：句法分析，将源语言句子分析为一棵句法结构树（短语结构树）；树到树的转换：递归地将源语言句子的句法结构树转换为目标语言句子的句法结构树，拼接叶结点得到译文。
- 树到串的代表模型，如Yang Liu(ACL2006)。其建模过程包括：在源语言端进行句法分析；在目标语言端不进行句法分析；从源语言端句法分析和词语对齐的语料库中抽取翻译规则；递归地将源语言句子的句法结构树转换为目标语言句子（树到串的转换）。
- 串到树的代表模型，如Galley et al.(2004, 2006)。其建模过程包括：在目标语言端进行句法分析（在源语言端进行不句法分析）；从目标语言端句法分析和词语对齐的语料库中抽取翻译规则并构造翻译模型；利用串到树转换规则，将源语言句子分析为一棵目标语言句法结构树，拼接叶结点得到译文。

深度学习方法解决机器翻译问题

2016年Google针对多语言翻译任务提出了一种新的解决方案，其对于不同语言间的翻译只需对输入的翻译句子进行处理而模型保持相同，即single model to translating multiple languages。更为强大的是其模型能够实现Zero-shot translation，一种类似推理的能力。对于human，如果你同时掌握中文和英文以及中文和日语间的相互翻译，那么你将很容易实现日语和英文间的翻译，然而对于机器却很难做到，但是在Google的这篇文章中却很好的实现了这一点，其称此为zero-shot translation。

Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation这篇文章中所使用的网络结构与Google 2016年所提出的GNMT网络结构大致相同 (Yonghui Wu et al., 2016) ,只不过对输入数据增加了一个针对目标语言的人工标记。但是需要注意的是，虽让其能在一定程度上实现zero-shot translation，但是相较于同族语言 (English、French)，对于不同族 (family) 的语言如Spanish与Japanese间的翻译，其BLUE score却很难令人满意。

GNMT(Google's Neural Machine Translation)

GNMT于2016年提出，其试图克服传统NMT存在的三个gaps：

- 网络训练和翻译速度较慢。尤其是对于翻译问题，由于网络参数众多，其速度远慢于phrased-based systems。
- 对于未登陆词或罕见词汇，其模型不够鲁棒。尽管由于大多数rare words为组织机构名或地名等词汇，可以通过简单的复制一定程度上缓解。然而，对于较深的网络其attention mechanism也将变得不稳定，无法达到较好的效果。
- 传统的NMT有时无法对input sentence给出完整的翻译结果，换言之其无法cover整个input，这样会导致翻译内容无法满足要求。

针对上述gaps，GNMT分别提出了一系列的解决方案，并一定程度上取得了不错的效果。具体而言其网络的结构设计如下图所示：

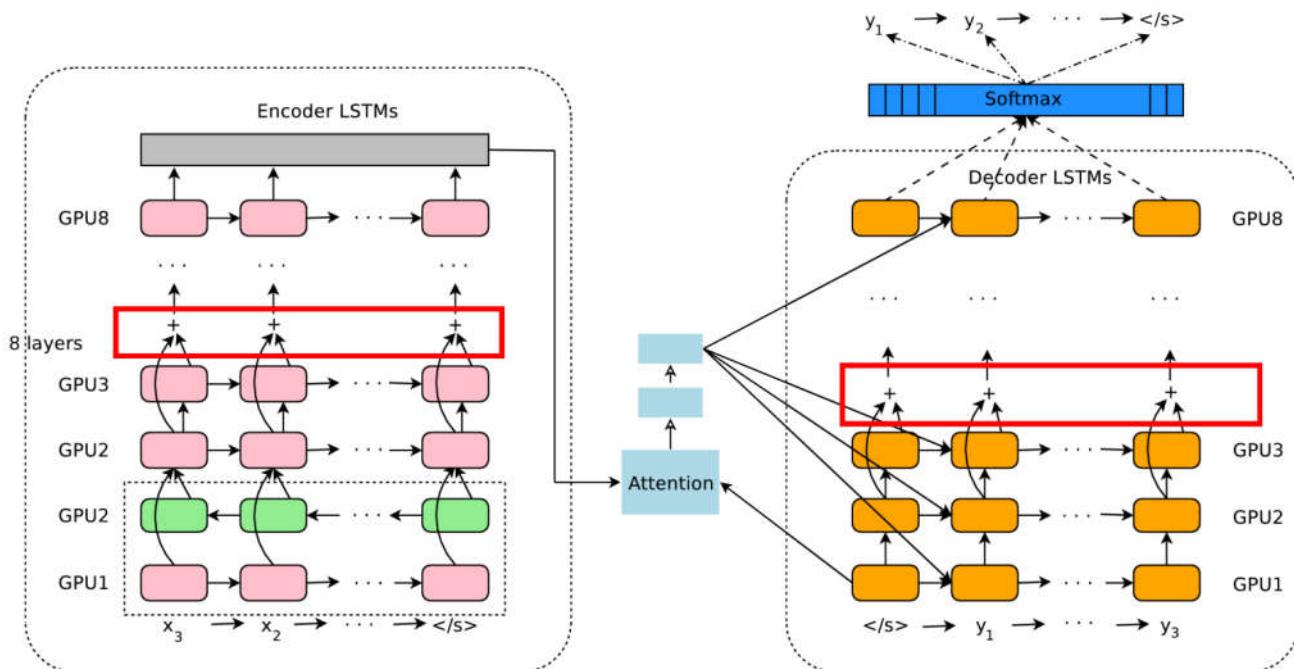


图20. GNMT网络结构

其具体细节如下：

- general:
 - (1) encoder和decoder共享embeddings；
 - (2) 网络使用多块GPU并行训练（包括数据并行和模型并行），训练策略为SGD结合Adam，Batch size一般为128；
 - (3) 采用GLUM Score代替简单的对数似然作为模型Training Criteria；

对于一般的网络其目标函数一般是最化对数似然：

$$O_{ML}(\theta) = \sum_{i=1}^N \log P_\theta(Y^{*^i} | X_i) \quad (2)$$

上式中， θ 为模型参数， x_i 为输入序列， Y^{*^i} 为预测值。然而观察上式，我们可以明显看出其只能反映翻译的整体水平，而并不能反映每个翻译结果的Blue Score的ranking，且模型的鲁棒性很差。因此这里引进 $r(Y, Y^{*(i)})$ 反映每一句子的得分并定义GLEU Score如下：

$$O_{RL}(\theta) = \sum_{i=1}^N \sum_{Y \in \mathfrak{Y}} \log P_\theta(Y^{*^i} | X_i) r(Y, Y^{*(i)}) \quad (3)$$

此外，为使模型结果更加稳定文中对 $O_{ML}(\theta)$ 、 $O_{RL}(\theta)$ 进行线性组合，如下：

$$O_{Mixed}(\theta) = \alpha O_{ML}(\theta) + O_{RL}(\theta) \quad (4)$$

上式中， α 为混合系数，文中取0.017。在具体的训练过程中，先选择 $O_{ML}(\theta)$ 使模型收敛，然后使用 $O_{Mixed}(\theta)$ 进一步提高翻的译准确性。

(4) 采用label smoothing loss (smoothing因子取0.1)；

- encoder:

(1) encoder采用8层LSTM，其hidden size为1024，第一层采用双向LSTM结构，其余层为单向结构；

Encoder第一层的双向连接结构如下：

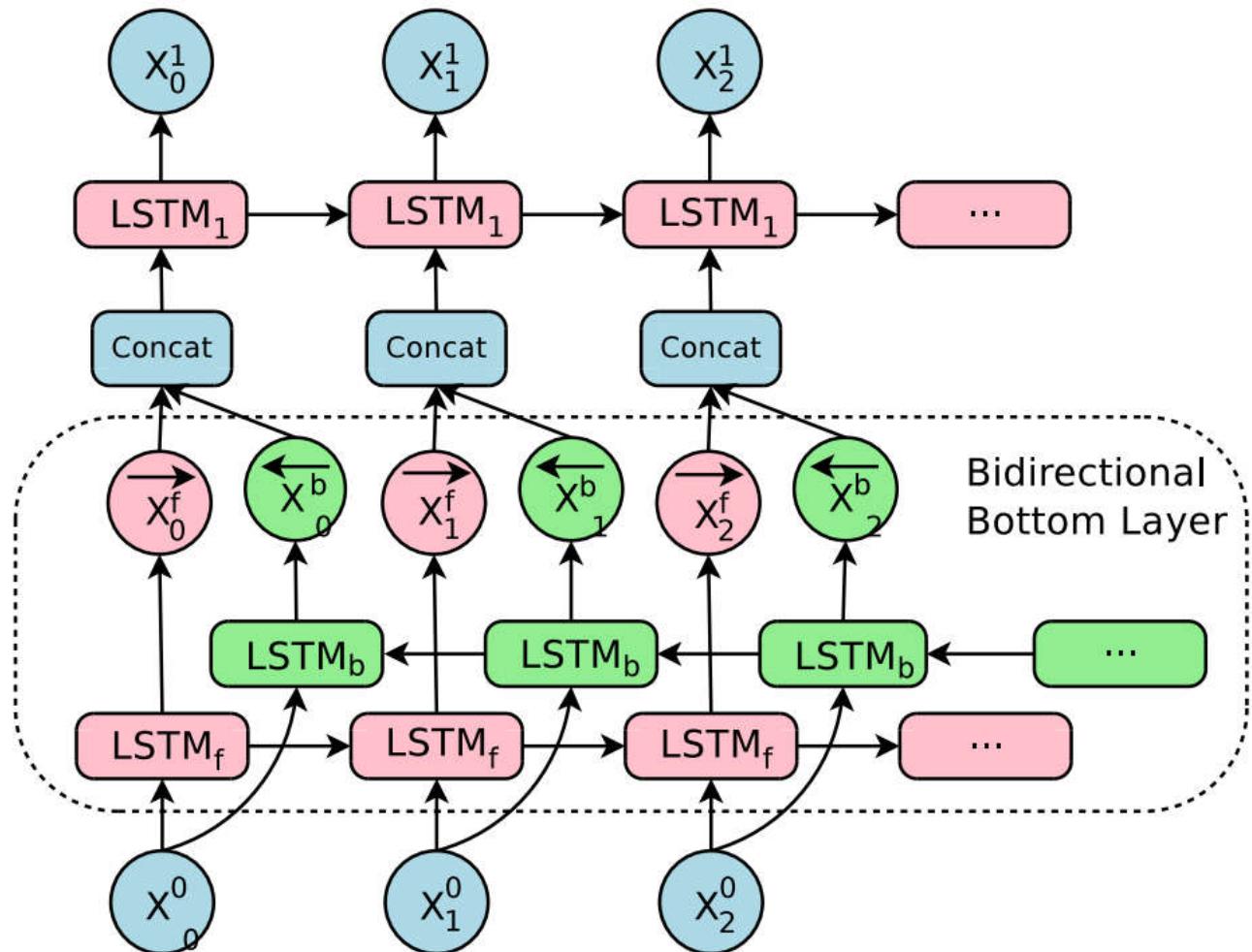


图21. 双向结构

文中设计的网络结构中只在Encoder的第一层使用了双向LSTM，其余的层仍然是单向LSTM，通过双向连接以更好的获得上下文信息。

(2) 从第三层开始采用残差连接；

其残差结构如下图所示：

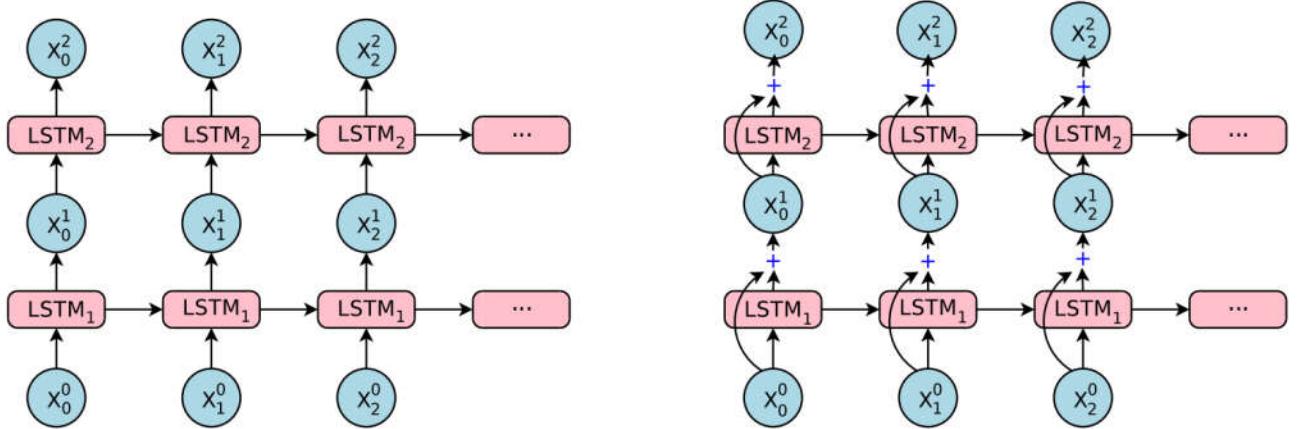


图22.残差结构

如上图所示，左边为普通的LSTM结构，右边即为残差连接结构。通过残差结构以提高网络的性能，训练更深的网络。考虑第*i*层和第*i+1*层LSTM，记网络参数分别为 W^i 和 W^{i+1} ，则在第*t*时刻对于普通LSTM及残差LSTM分别有：

(a) normal LSTM:

$$\begin{aligned} c_t^i, m_t^i &= LSTM_i(c_{t-1}^i, m_{t-1}^i, x_t^{i-1}; W^i) \\ x_t^i &= m_t^i \\ c_t^{i+1}, m_t^{i+1} &= LSTM_{i+1}(c_{t-1}^{i+1}, m_{t-1}^{i+1}, x_t^i; W^{i+1}) \end{aligned} \quad (5)$$

(b) residual LSTM:

$$\begin{aligned} c_t^i, m_t^i &= LSTM_i(c_{t-1}^i, m_{t-1}^i, x_t^{i-1}; W^i) \\ x_t^i &= m_t^i + x_t^{i-1} \\ c_t^{i+1}, m_t^{i+1} &= LSTM_{i+1}(c_{t-1}^{i+1}, m_{t-1}^{i+1}, x_t^i; W^{i+1}) \end{aligned} \quad (6)$$

上式中， x_t^i 为*t*时刻*LSTM_i*的输入， m_t^i 和 c_t^i 分别为hidden states及memory states。

- decoder:

(1) 8层单向LSTM且为全连接结构，其hidden size为1024；

(2) 至第三层开始采用残差连接；

- attention:

(1) 对Bahdanau attention进行标准化处理；

(2) encoder最后一层的输出于decoder第一层的输入结合，然后在当前timestep从新计算权值及context；

其中context的计算公式如下：

$$\begin{aligned} s_t &= AttentionFunction(y_{t-1}, x_t), \quad \forall t, 1 \leq t \leq M \\ p_t &= \exp(s_t) / \sum_{t=1}^M \exp(s_t) \end{aligned}$$

$$a_i = \sum_{t=1}^M p_t x_t \quad (7)$$

其各参数代表含义均与Attention Mechanism相同。

- inference:

- (1) 采用beam search, 其beam size为默认值5;
- (2) 引入惩罚项, 并对长句子进行normalization处理;

此外, 文中还使用了Wordpiece Model及Quantizable Model and Quantized Inference的策略进一步提高模型的性能。

对于未登录词的翻译问题 (OOV, out-of-vocabulary), 由于这些未知词通常是日期, 人名, 地名等, 所以一个简单的方法就是直接复制这些词, 而GNMT中, 其采用了wordpiece model, 如下:

- Word: Jet makers feud over seat width with big orders at stake
- wordpieces: _J et _makers _fe ud _over _seat _width _with _big _orders _at _stake

图23. wordpiece

对比word和wordpiece我们可以看到“Jet”被拆分为“_J”和“_et”，其中“_”表示一个单词的开始。通过上述处理其在单词处理的灵活性和准确性上取得了一个很好的均衡, 同时也提高了翻译的准确率和速度。

由于在残差的结构中 c_t^i, x_t^i 值均较小, 为了减少错误的累积同时使较深的网络较快的收敛, 文中将其约束在 $[-\delta, +\delta]$ 的范围内, 具体公式参阅原文, 这里就不展开了。

综上所述, GNMT主要包括四个关键点:

- Key1. WordPiece Model有效的解决了未登录词以及心态学丰富语料的问题, 提高了翻译的质量及速度;
- Key2. 模型及数据的并行处理有效提高了Seq2Seq NMT的训练速度;
- Key3. Quantizable Model and Quantized Inference的使用加速了翻译的速度;
- Key4. 其他的trick, 如length-normalization, coverage penalties, similar均对模型的性能做出了贡献。

总之通过上述一系列操作, 最后GNMT在多个流行语种间的翻译性能 (速度、准确性和鲁棒性) 得到了很高的提升 (文中写到翻译错误率减少了近60%) 。

GMNMT (Google's Multilingual Neural Machine Translation)

GMNMT被在多语言翻译任务中被描述为一种优雅、简单的方法。其使得多语言间的transfer learning以及Zero-shot translation成为了可能。

具体来说, 该模型的结构与GNMT完全相同, 只不过对输入数据做了些简单处理。因此对于语料信息较少(数据较少) 的语种 (小语种) 由于模型相同, 因此其也能得到不错的翻译结果。

How are you? -> ¿Cómo estás?

It will be modified to:

<2es> How are you? -> ¿Cómo estás?

图24. GMNMT语料处理

GMNMT在对多语言进间进行翻译时，一般其会将目标语言映射至一个interlingua (“中间语言”)。如下图所示：

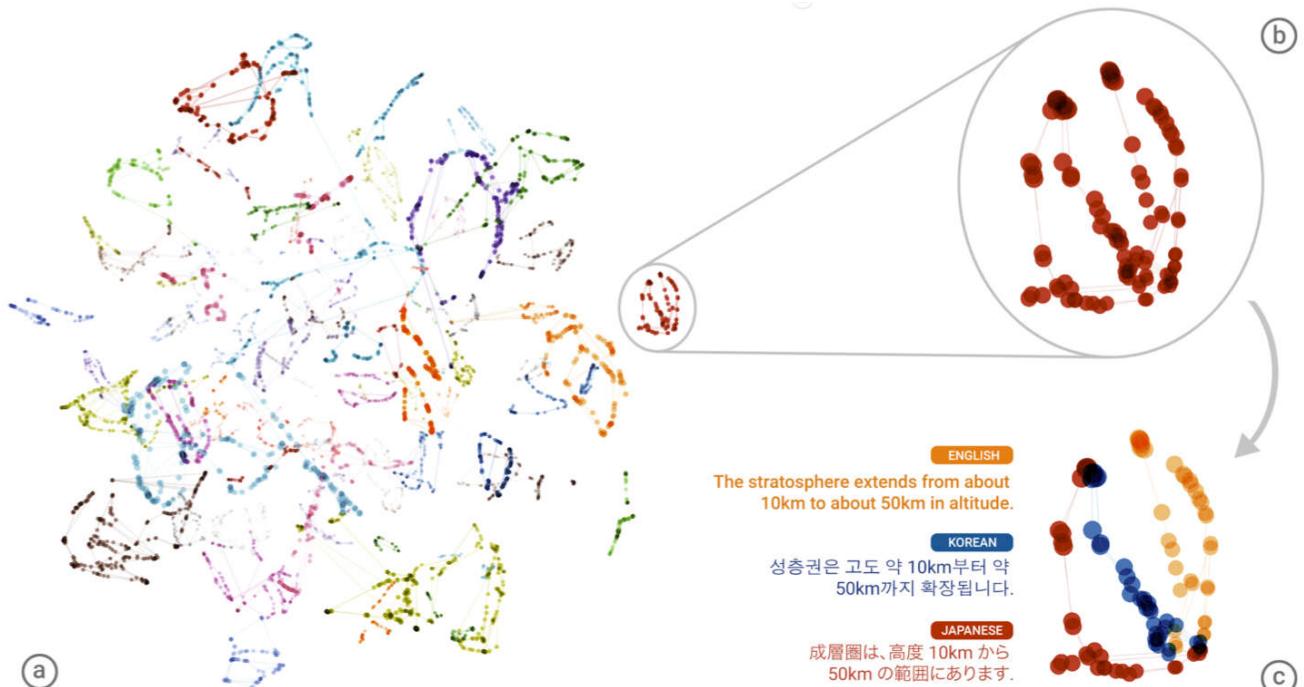


图25. GMNMT语料处理

上图中将74个独立的语句（同一句话（语义相同）但是不同语言）在循环网络里把每一步的表征嵌入到低维空间中，最后总会汇入到一个点上。此外文中作者还进行了一些有趣的尝试，如将多语言进行混合、加权翻译等。总而言之，Google16年的这两篇文章可以认为在NMT中取得了突破性的进展。

Transformer Attention

Transformer模型最早是由Google于2017年在“Attention is all you need”一文中提出，在论文中该模型主要是被用于克服机器翻译任务中传统网络训练时间过长，难以较好实现并行计算的问题。后来，由于该方法在语序特征的提取效果由于传统的RNN、LSTM而被逐渐应用至各个领域，如GPT模型以及目前Google最新出品的新贵Bert模型等。相较于传统的RNN、LSTM或注意力集中机制，Transformer模型已经抛弃了以往的时序结构（这也是其为何能够很好的并行，且训练速度较快的原因），更准确的来说其实际上是一种编码机制，该编码同时包括了语义信息（Multi-Head Attention）和位置信息（Positional Encoding）。（其实机器翻译问题某种程度上可以看作是编码问题，即将一种语言编码为另一种语言，统计机器翻译的核心思想）。下面详细介绍该模型。

对于机器翻译这类Seq2Seq问题，该模型从整体结构来看仍使用了Encoder-Decoder框架。而其Encoding component由6个encoder共同堆叠而成，而每个Encoder中又包括Self-Attention和Feed Forward Neural Network两层网络结构。Decoding component的组成基本如此，只不过在Self-Attention和Feed Forward间添加了一层Encoder-Decoder Attention用于关注Encoder的编码信息（与Encoder-Decoder的C向量类似）。其结构如下所示：

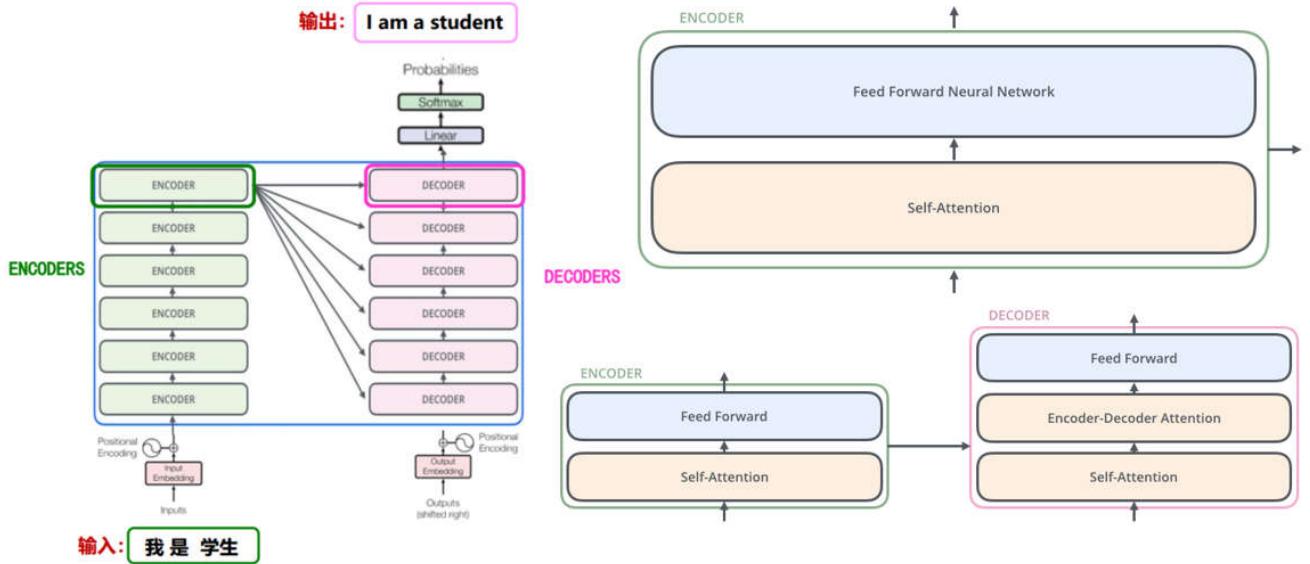


图26. 模型结构

现在具体分析模型的内部细节。

1. Encoders

(1) Self-Attention

对于encoder端的输入信息首先我们将文本字符进行编码为向量（512维）（Embedding操作），然后送入Self-Attention层。在Self-Attention层中对于每一个Embedding向量 X ，网络定义三个矩阵 W^Q 、 W^K 、 W^V （其中矩阵的维数为64，经过计算后维度大大减小，计算复杂度明显降低），分别对每个输入计算三个向量：queries、keys and values，以得到输入句子中的每个词与其他词之间的关系，以确定在某个位置编码特定单词时，应该将注意力集中于输入句子的其他部分。

$$\begin{aligned}
 Q &= X * W^Q \\
 K &= X * W^K \\
 V &= X * W^V \\
 Z &= softmax\left(\frac{Q * K^T}{\sqrt{d_k}}\right) * V
 \end{aligned} \tag{8}$$

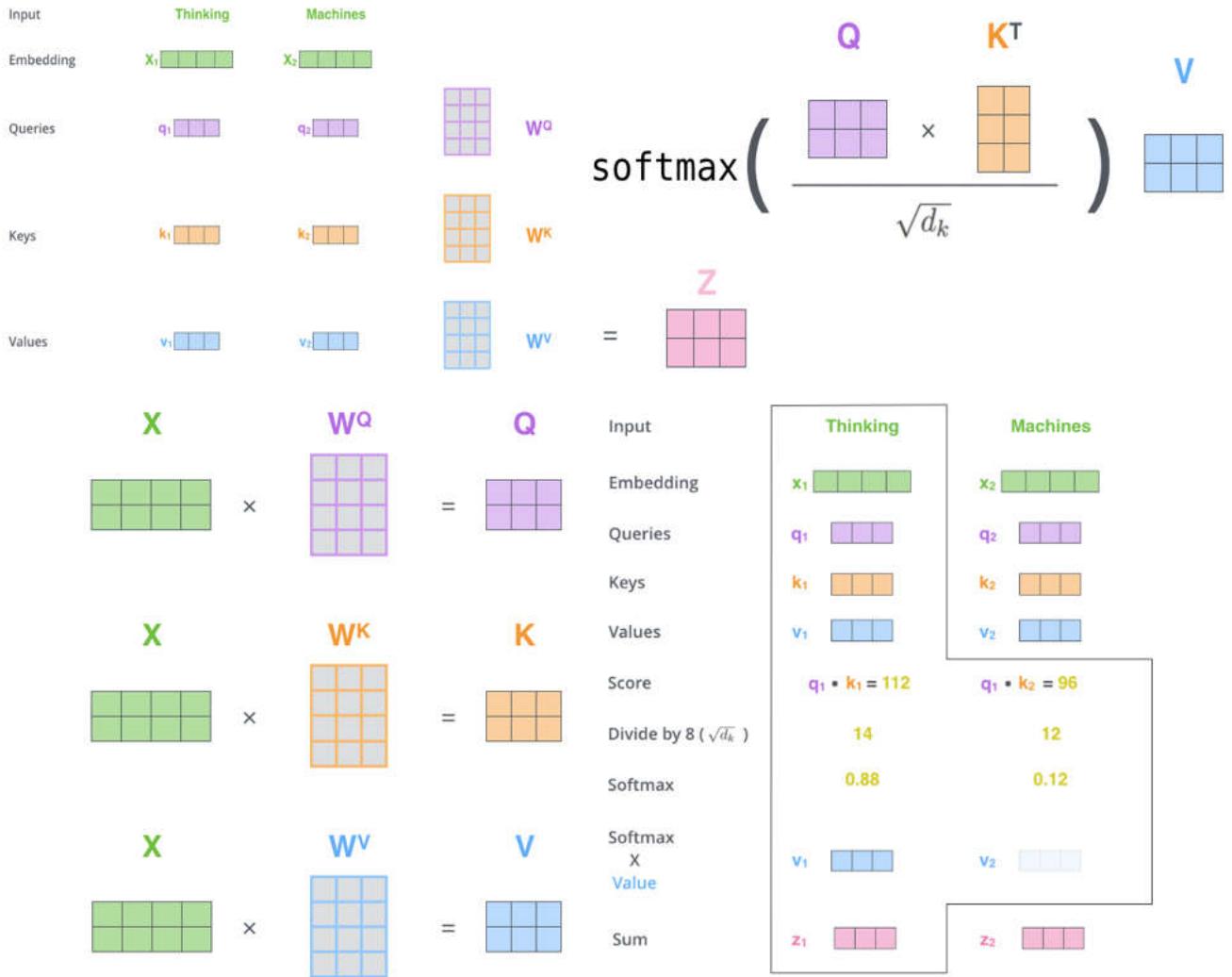


图27. self attention操作

注意到矩阵 Q 、 K 、 V 中的每一行即包含了每一个单词的信息，而通过矩阵间的运算我们能和好的融合每个输入单词间潜在的语义信息，最后经过softmax操作得到此位置上每个单词被关注程度的打分（权重），其中除以 $\sqrt{d_k}$ ， d_k 取较大值，防止 $Q * K^T$ 较大，进入Softmax饱和区。最后 Z 的计算即为简单的加权求和。从这里我们可以明显看出此时的Attention机制已经不同于之前的操作，这里主要是根据各词向量间的矩阵运算实现Self-Attention（输入句子中各个词向量间的相互运算，不涉及输出，故为“Self”），通过Self-Attention其实网络学习到的是句法语义结构。

(2) Multi-header attention

该文章在Self-Attention的基础上进而设计了Multi-header attention以提高网络的性能。即将多个Self-Attention得到的 Z 矩阵进行concat的操作然后融合。具体来说文中主要利用了8个head进行拼接，然后进行矩阵乘法运算，融合信息。如下：

$$Concat(Z_0, Z_1, \dots, Z_7) * W^O = Z \quad (9)$$

即：

$$\begin{aligned} MultiHead(Q, K, V) &= Concat(head_1, \dots, head_h)W^O \\ head_i &= Attention(QW_i^Q, KW_i^K, VW_i^V) \\ \text{where } W_i^Q &\in R^{d_{model} \times d_k}, W_i^K \in R^{d_{model} \times d_k}, W_i^V \in R^{d_{model} \times d_k}, W^O \in R^{hd_{model} \times d_k} \end{aligned} \quad (10)$$

- 1) This is our input sentence* X
- 2) We embed each word* R
- 3) Split into 8 heads. We multiply X or R with weight matrices W_0^Q, W_0^K, W_0^V
- 4) Calculate attention using the resulting $Q/K/V$ matrices
- 5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer

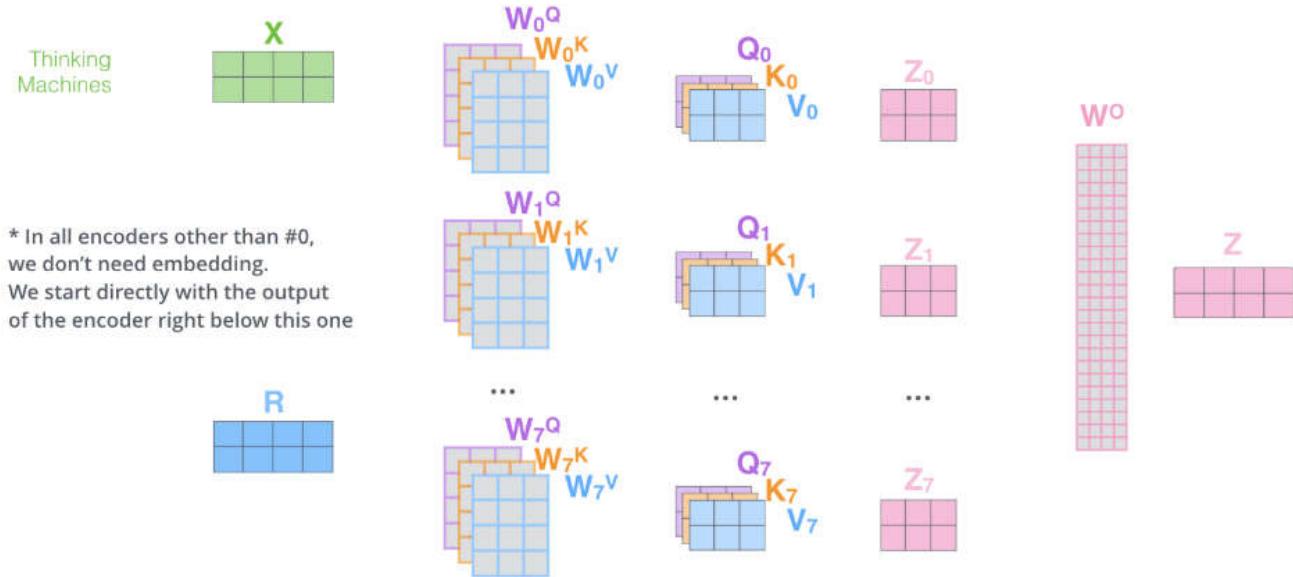


图28. Multi-header attention结构

相较于Self-Attention更多的关注于个单词于其他单词间的联系而言，通过Multi-header attention机制扩展了模型对句子不同位置关注的能力同时扩展了模型的“表征子空间”。

(3) Positional Encoding

上述无论是Self-Attention，又或是Multi-header attention结构其本质上都是词与词间的相互计算，即词袋模型，而未对每个词的位置先后顺序进行建模。为克服这一缺点文中引入位置编码。其具体操作时在Encoder的底端Embedding对应位加上位置编码（维度与embedding一致，512，以使得求和可行）。其中位置编码可以由网络学习而来，亦可以通过设计编码函数进行生成。文中作者使用正弦曲线生成编码，以使得模型对不同输入句子长度的编码具有扩展性。

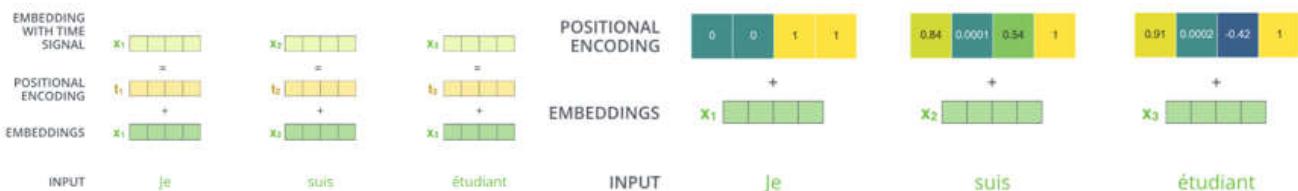


图29. 位置编码操作

(4) The Residuals

在每一个Encoder结构中，作者还设计了Normalize操作（即加和完成后即对其进行Normalization）和残差结构（因该是考虑到6个Encoder网络较深，防止梯度消失），如下图所示。

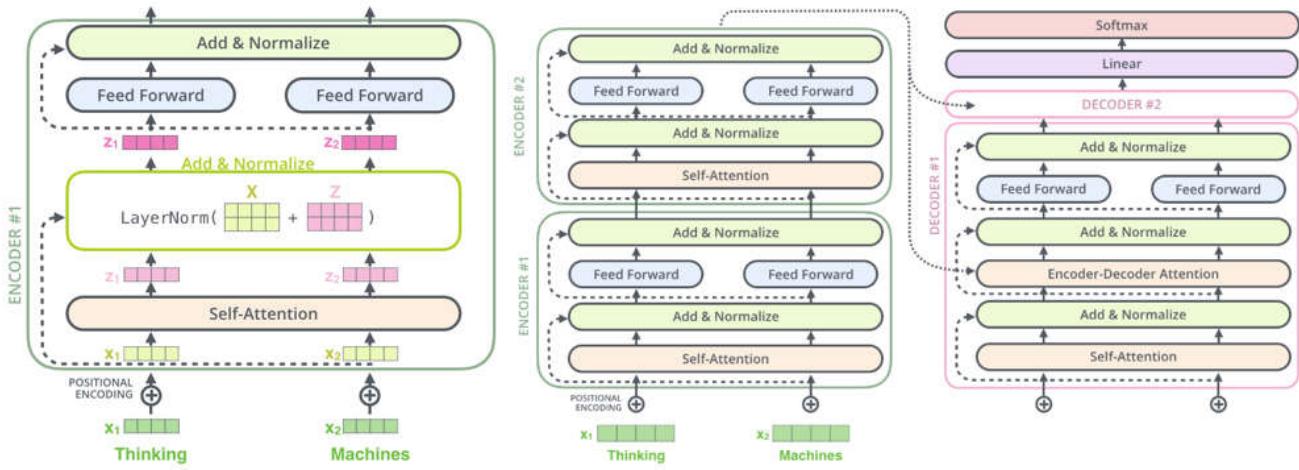


图30. 残差结构

最终网络的Encoders端如下图所示：

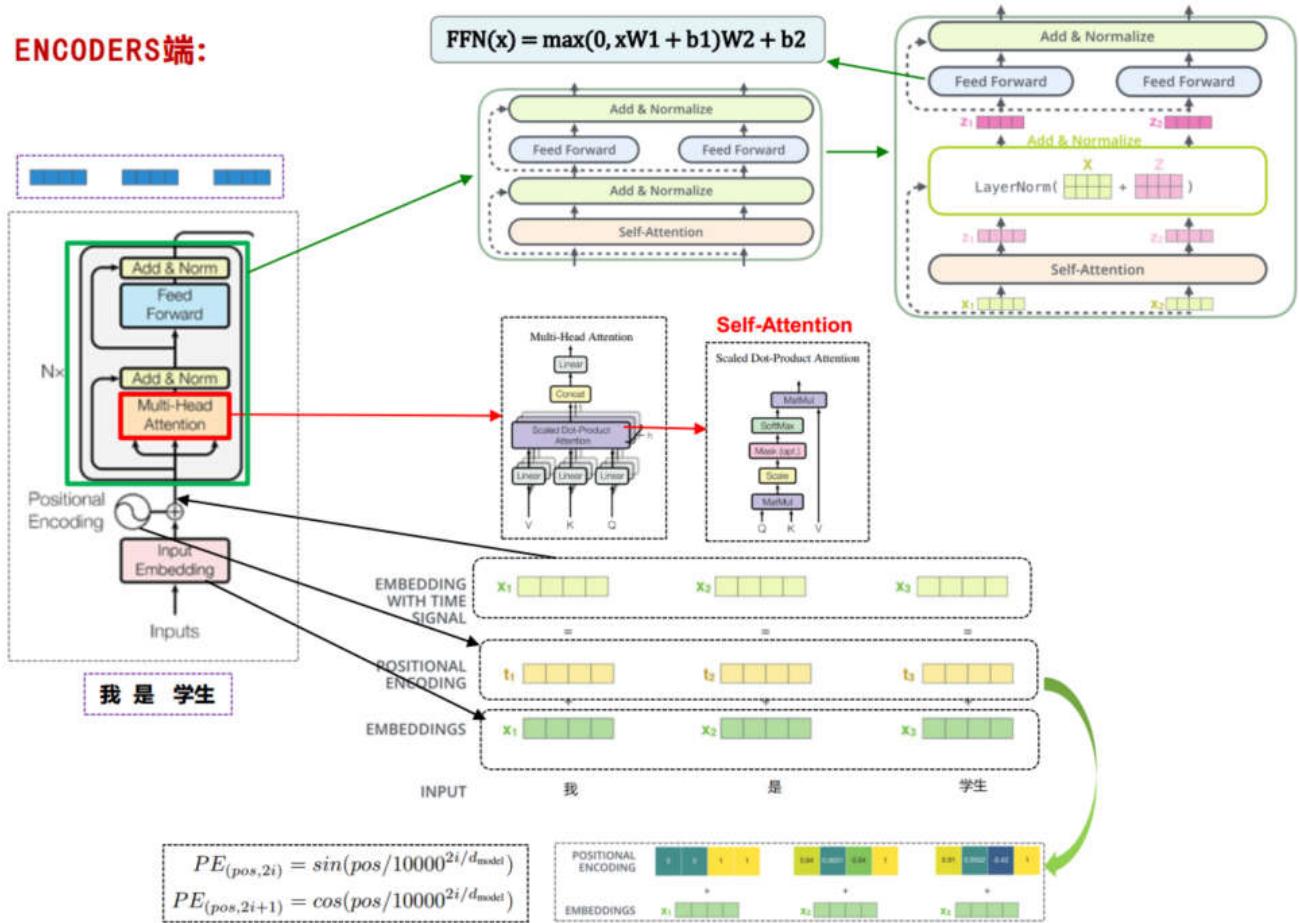


图31. Encoders结构

2. Decoders

综上所述，由Encoders端产生的 K 、 V 矩阵将送入至Decoders端，同时Decoders端结合上一步的翻译结果共同给出此时的翻译结果，而稍与Encoder不同的是，其self-attention layer将只关注前文信息而将后文输出置为-inf。最后，Decoder stack外接一个Linear和Softmax Layer，将向量对应为最后的翻译单词。其中linear为全连接层，其维度即为翻译目标语言语料库包含所有的次数目。再外接Softmax层即可得到每一输入词（假设中文）翻译为另一种语言（英文）中每一个词的概率。

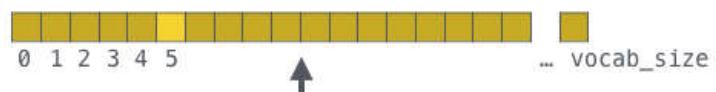
Which word in our vocabulary
is associated with this index?

am

Get the index of the cell
with the highest value
(argmax)

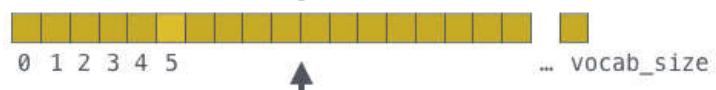
5

log_probs



Softmax

logits



Linear

Decoder stack output

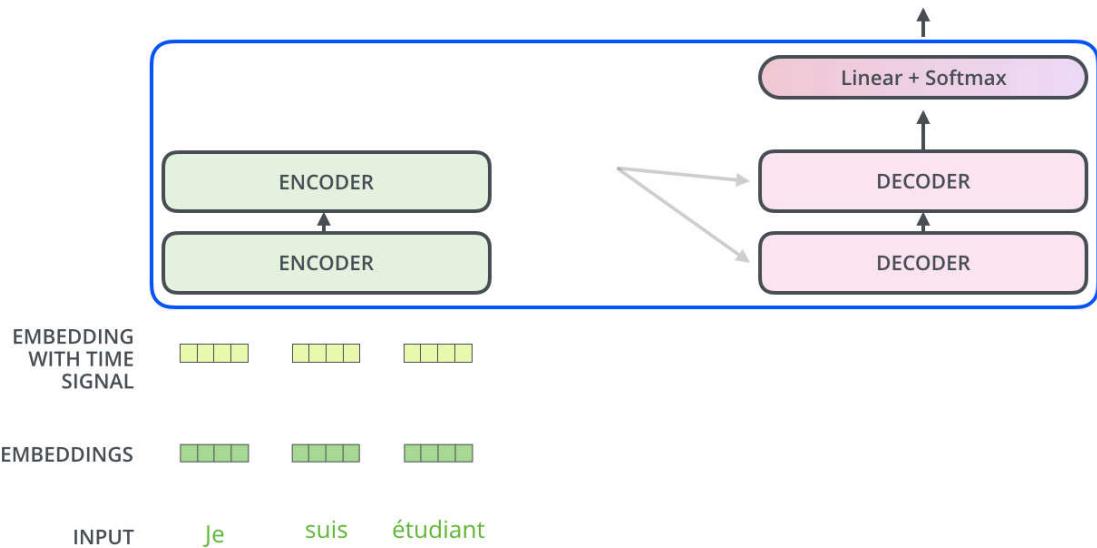


图32. linear-softmax结构

Decoders的具体翻译过程如下图所示：

Decoding time step: 1 2 3 4 5 6

OUTPUT



Decoding time step: 1 2 3 4 5 6

OUTPUT

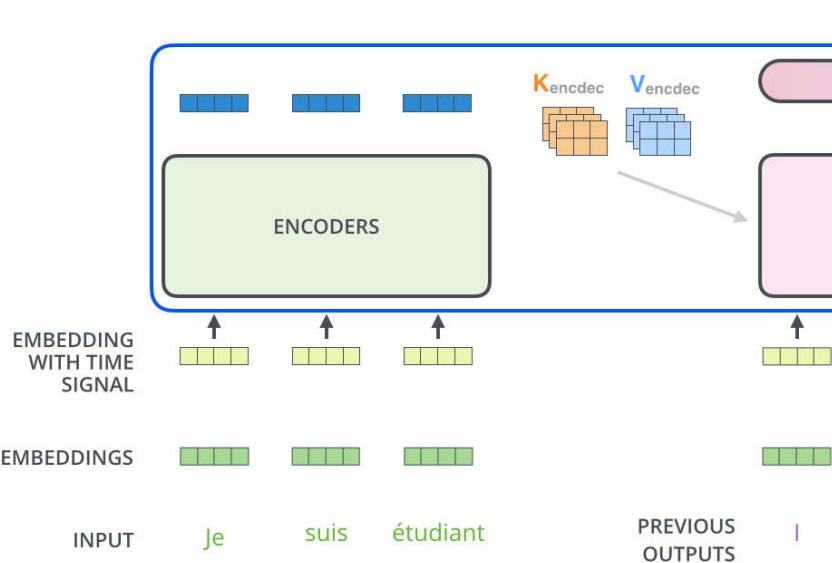


图33. Decoders翻译过程

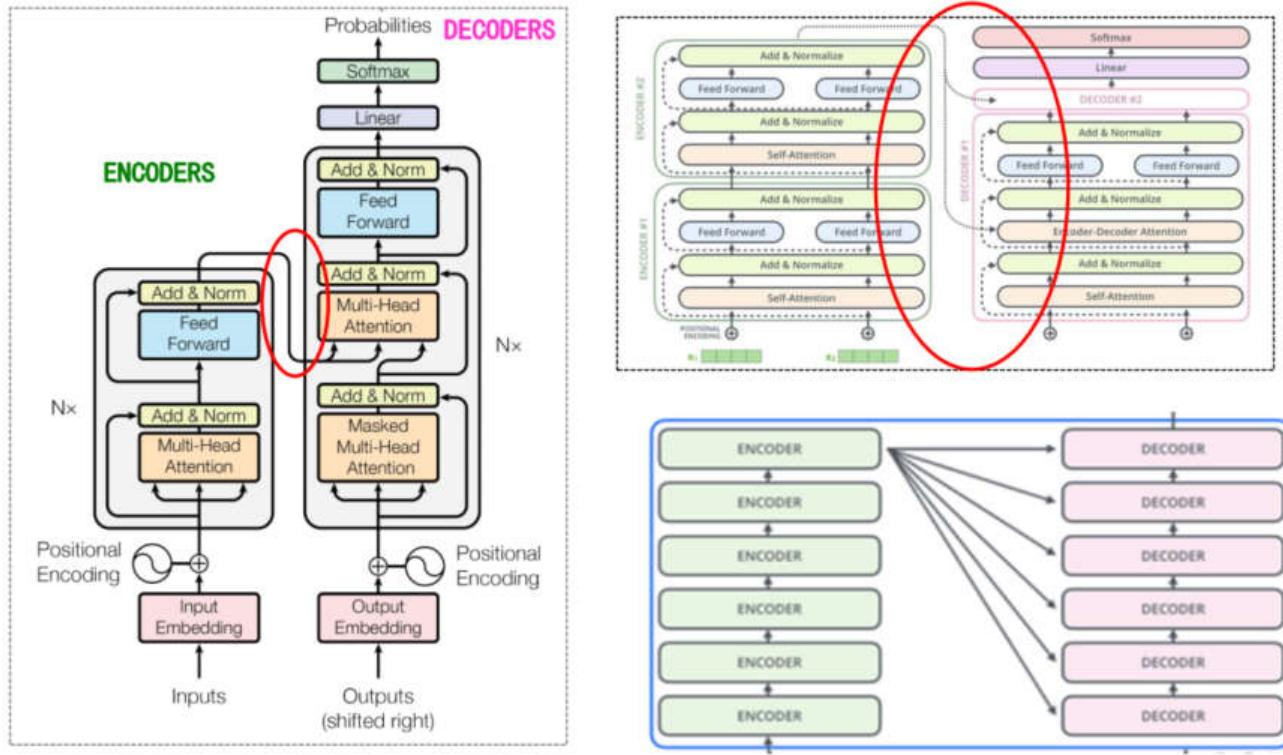


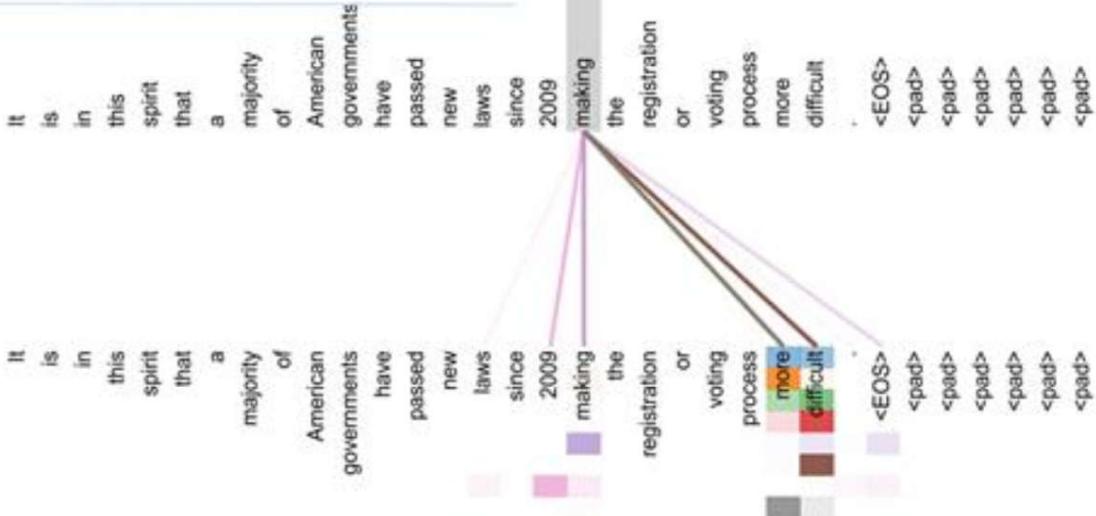
图34. Encoder-Decoder连接结构

3. Experiments

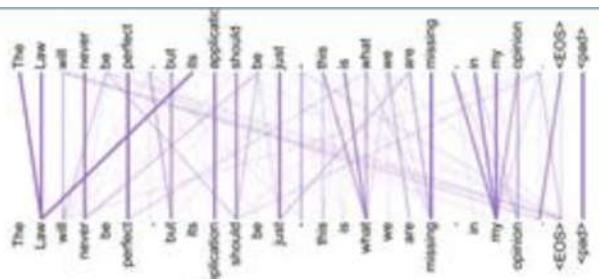
最后文中对于Self-Attention及Multi-head self attention给出了可视化的结果，如下：

Self-attention

Different colors represent different heads



One head



- Many of the attention heads exhibit behaviour that seems related to the structure of the sentence.

Two head

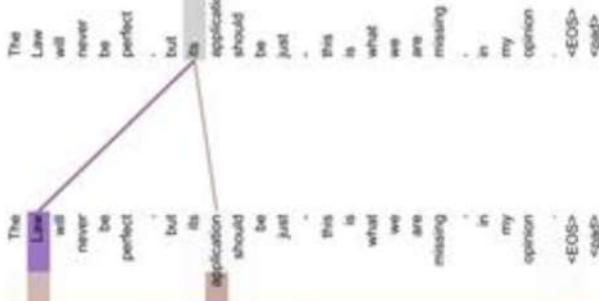
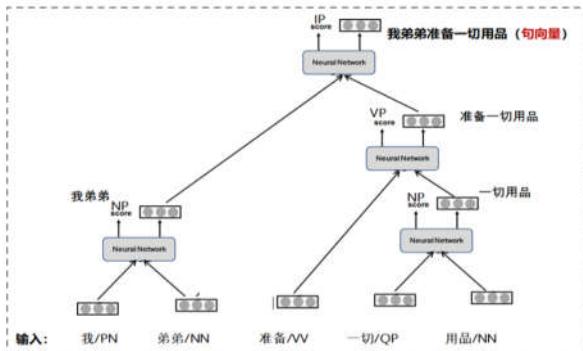


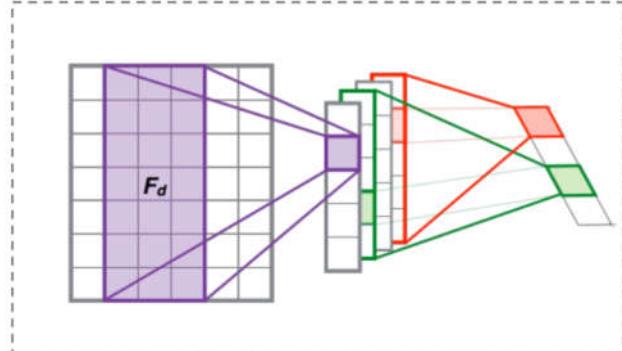
图35. Self-Attention与Multi-Head Self Attention结果可视化

从Self-Attention的可视化结果可以明显看出，对于长句子中相隔较远的且存在一定关联的词，Self-Attention也能很好的学习到其相互间的依赖关系（making...more difficult）。而Multi-Head Self Attention其能够从不同的表示子空间里学习相关信息（在two head和one head的比较中，可以看到one head中"its"这个词只能学习到"law"的依赖关系，而two head中"its"不仅学习到了"law"还学习到了"application"依赖关系）。

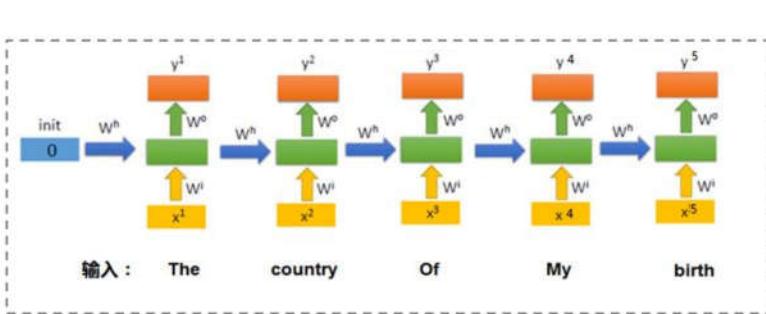
最后对比各网络结构间句向量的表达形式，如下：



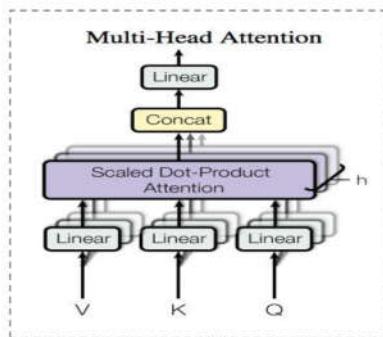
RvNN 句向量



CNN 句向量



RNN 句向量



Multi-Head Self Attention 句向量

图36. 不同网络结构句向量的构建方法

由图36可以看出，RNN其实就是序列中每个词向量的拼接。而RvNN通过构建句法树考虑词与词间的依存关系，构建词向量。CNN通过滑动窗口实现。而Multi-Head Self Attention即使Self-Head Attention的Concat。

Demo

这里我们分别使用Attention Mechanism和Transformer两种模型完成中译英的机器翻译任务。考虑到机器的性能因素，这里我们只选取6834个中英平行语对进行训练（将6800组作为训练语料，34组作为测试语料），同时定性分析两种模型的收敛情况，而不将最后输出的准确率作为比较对象。

Environment Requirements

The model must be run on Python ≥ 3.5 and Tensorflow ≥ 1.10 (one-point-ten).

Hyper-parameters setting

(1) Attention mechanism

- SRC_TRAIN_DATA = train_cn_path # 源语言输入文件路径（中文）。
- TRG_TRAIN_DATA = train_en_path # 目标语言输入文件路径（英文）。
- CHECKPOINT_PATH = checkpoint_path # checkpoint保存路径。
- HIDDEN_SIZE = 1024 # LSTM的隐藏层规模。
- DECODER_LAYERS = 2 # 解码器中LSTM结构的层数。其中编码器固定使用单层的双向LSTM。
- SRC_VOCAB_SIZE = 4000 # 源语言词汇表大小（中文）。
- TRG_VOCAB_SIZE = 10000 # 目标语言词汇表大小（英文）。
- BATCH_SIZE = 100 # 训练数据batch的大小。
- NUM_EPOCH = 200 # 使用训练数据的轮数。

- KEEP_PROB = 0.8 # 节点不被dropout的概率。
- MAX_GRAD_NORM = 5 # 用于控制梯度膨胀的梯度大小上限。
- SHARE_EMB_AND_SOFTMAX = True # 在Softmax层和词向量层之间共享参数。
- MAX_LEN = 50 # 限定句子的最大单词数量。
- SOS_ID = 1 # 目标语言词汇表中的ID。

(2) Transformer

```
batch_size = 32 # alias = N lr = 0.0001 # 学习率, 原文中学习率在global step中并未按指数衰减 logdir =
path_log # 日志保存路径 maxlen = 50 # 句子最大长度 (词数目) . alias = T min_cnt = 2 # 语料中, 低频词最
num_epochs = 200 # 迭代数 num_heads = 8 # head数目, 原文中为8 dropout_rate = 0.1 # the rate you want
to drop out. sinusoid = False # 位置编码, True使用正弦函数, false使用embedding, 原文中为正弦函数
```

Results

最后的结果如下图所示：

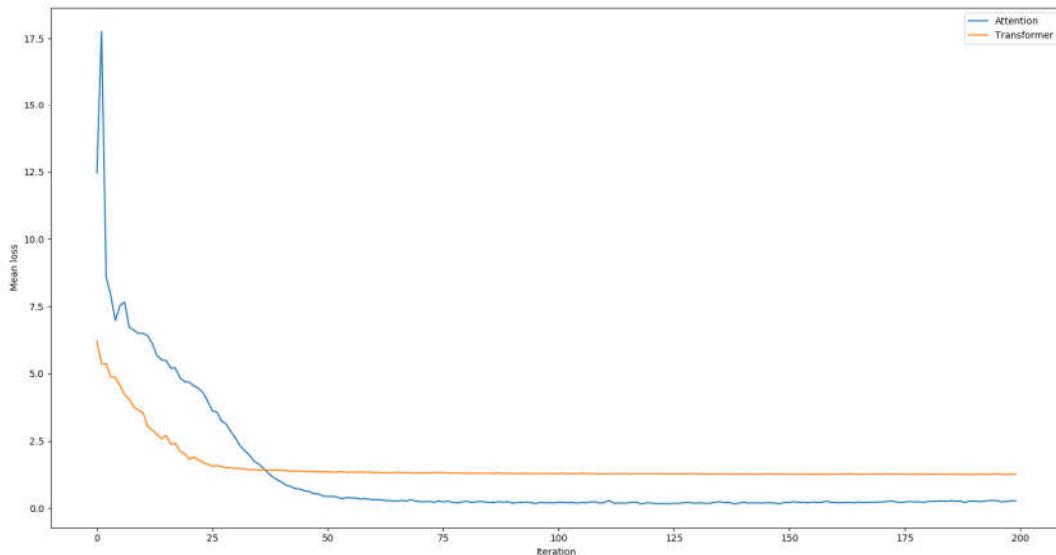


图37. mean loss

从上图可以明显看出，对比Attention，Transformer收敛更快（约30 epochs已经收敛），且更平滑。在实际训练过程中Transform耗时更短（Transformer是以克服序列建模并行困难，训练耗时为motivation而提出的）。由于训练数据较少这里我们不对mean_loss进行比较，但实际应用中Transformer的Blue值要优于Attention。

这里我们也可以简单对比下在小语料下人工智障的翻译结果：

```
source: 众所周知，中美两国之间的军事磋商与合作，在布什政府上台后一直陷于半停滞状态。
target:as is known to all , military consultation and cooperation between china and the united states has been at a semi-standstill since the bush administration took office .
translate: <sos> as is known to all , military consultation and cooperation between china and the united states has been at a <unk> since the bush administration took office . <eos>
source: 质量认证和合格评定作为质量管理和贯彻标准的新兴手段，正逐步为世界各国所重视。
target:quality certifications and assessments , as a new means for quality control and implementation of criteria , have drawn the ever-increasing attention of all countries in the world .
translate: <sos> quality <unk> and assessments , as a new means for quality control and implementation of criteria , have drawn the ever-increasing attention of all countries in the world . <eos>
source: 各国文明的多样性，是人类社会的基本特征，也是人类文明进步的动力。
target:the diversity of the civilizations of all countries is the basic feature of human society and is the driving force of progress in human civilization .
translate: <sos> the cultural diversity of various countries is a basic characteristic of human society and also a driving force for the progress of mankind 's civilizations . <eos>
```

图38. Attention translation

```
- source: 众所周知 中美两国之间的军事磋商与合作 在布什政府上台后一直陷于半停滞状态  
- expected: as is known to all, military consultation and cooperation between China and the United States has been at a standstill since the Bush administration took office  
- got: as all along with the Bush administration felt three sinous relations were used to be <UNK> and Bush Jr's policy toward the United States  
  
- source: 质量认证和合格评定作为质量管理和贯彻标准的新手段 正逐步为世界各国所重视  
- expected: quality certifications and assessments as a new means for quality control and implementation of criteria have drawn the everincreasing attention of all countries in the world  
- got: the member states should attach importance to ideological front work and make joint efforts to strengthen their role for foreign affairs and use of opportunities  
  
- source: 各国文明的多样性 是人类社会的基本特征 也是人类文明进步的动力  
- expected: the diversity of the civilizations of all countries is the basic feature of human society and is the driving force of progress in human civilization  
- got: the cultural diversity of various countries is a basic characteristic of human society and also a driving force for the progress of mankind's civilizations
```

图39. Transformer translation

2017年同年，Facebook也提出了Convolutional Sequence to Sequence Learning中其不同于以往RNN、LSTM的方法，通过CNN的网络结构实现seq2seq learning，并且在英-德和英-法翻译中其性能远高于GNMT，且更容易实现并行。最近Google又开源了Bert，其实现了上百种语言多种NLP任务的Pre-training，在Machine Translation上也有不错的表现，期待其能有更大的突破。

Reference

- [1] Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. In International Conference on Learning Representations (2015)
- [2] Cho, K., van Merriënboer, B., Gulcehre, C., Bougares, F., Schwenk, H., and Bengio, Y. (2014a). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Proceedings of the Empirical Methods in Natural Language Processing (EMNLP 2014).
- [3] Cho, K., van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014b). On the properties of neural " machine translation: Encoder–Decoder approaches. In Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation.
- [4] Chung, Junyoung; Gulcehre, Caglar; Cho, KyungHyun; Bengio, Yoshua (2014). "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling".
- [5] Schuster M, Paliwal K K. Bidirectional recurrent neural networks[J]. IEEE Transactions on Signal Processing, 2002, 45(11):2673-2681.
- [6] Vaswani A , Shazeer N , Parmar N , et al. Attention Is All You Need[J]. 2017.
- [7] Yonghui Wu, Mike Schuster, and Zhifeng Chen et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144v2.
- [8] Johnson M , Schuster M , Le Q V , et al. Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation[J]. 2016.