3.5.5.4 循环神经网络（RNN）& 长短时记忆（LSTM）

在阅读时，您不会去孤立地去理解每个词，而忽略之前的内容；当您在观看视频时，您也不会孤立地只看其中的一帧，而需要考虑之前相关帧所组成的序列；当在对金融市场进行预测时，需要分析昨天、前天和更长时间的历史数据。然而，令人吃惊的是：如此常见的场景，大部分的神经网络竟然无法实现，原因很简单，因为它们是没有记忆的！本节将介绍一种的重要深度学习方法 — ***循环神经网络***（Recurrent Neural Network，RNN）与它的一种特殊形式 — **长短时记忆**（Long Short Term Memory，LSTM），专门用于序列数据的分析处理，例如文本、语音、股价等任何依赖于之前元素的时间序列。

如果说CNN是在图像处理领域的法宝，那么RNN就是自然语言处理（Natural Language Processing，NLP）领域的利器。自人工智能诞生以来，NLP一直是AI的重要研究领域，由于其研究难度大，被称为人工智能王冠上的明珠。NLP涉及的范围很广，拼写检查、联想输入法、关键字搜索、情绪分析、文档分类、机器翻译、语音识别、对话系统和复杂问题回答等都属于NLP的范畴。

在NLP领域中，语言学家们经常使用***语言模型***（Language Model）根据一个句子的前面部分，预测接下来最可能出现的词，这是一个计算句子概率的概率模型。在语音识别中，声学模型输出的结果是若干候选词或句子，例如，声学模型输出2个候选句子，算法要从中选择一句可能性更高的语句作为输出：1）你现在在干什么？2）你现在在干失眠？显然第一句话出现的概率要远大于第二句，于是系统就选择第一句作为最终的识别结果。进行图像文本识别的光学字符识别（Optical Character Recognition，OCR）也可以应用语言模型来提高识别率。SAP正在研发一款发票自动识别应用正是将CNN与这种技术结合。

在RNN诞生之前，语言模型主要采用基于统计的模型N-Gram，N是一个不太大的自然数，经常取值如2、3、4等。如果采用2-Gram（也称bi-Gram），语言模型只考虑前一个字，在上面的例子中，系统会在语料库中搜索“干”后面最可能出现的词，显然“干什么”比“干失眠”的可能性要大很多。在这个例子中2-Gram已经够用了，但是大部分情况下，2-Gram的模型还是过于简单了，比如：我 昨天 上学 迟到 了，老师 批评 了___。要估计这个词，2-Gram就不够用了， 答案在8个字之前，需要7-Gram才能解决。然而，实际上这难以实现，因为：首先，会有一些情况7-Gram也不够，我们需要能处理任意长度的的句子；其次，随着N的增加，所需要占据的存储空间指数增长；第三，答案在N（N很大）个词以外的情况太稀少，语料库中也不一定存在这样的情况。 可见基于统计的语言模型有其局限性，这时，RNN的必要性就显现了。

图3-101展现了RNN及其展开后的形式，两种表示方法完全等效。图中方块A代表神经网络的模块，被称为**“单元”**（Cell），可以把RNN想象为不断的复制出来并展开的单元，每个“单元”对输入序列中对象进行***相同的***操作。RNN“单元”通过**隐藏状态**（Hidden State，h）来“记忆”和传递之前所有“看过”的内容。在时刻t，单元A读取输入 x_t和前一时刻的隐层状态h_{t-1}，经处理后输出隐层状态h_t，h_t会被传递到了下一时刻（$t+1$），**在所有时刻，单元A都使用同一套神经网络权重。**注意到：如果把图中RNN指向自身的循环箭头去掉，该RNN就退化为只含一个隐层的（全联接）前馈式神经网络。

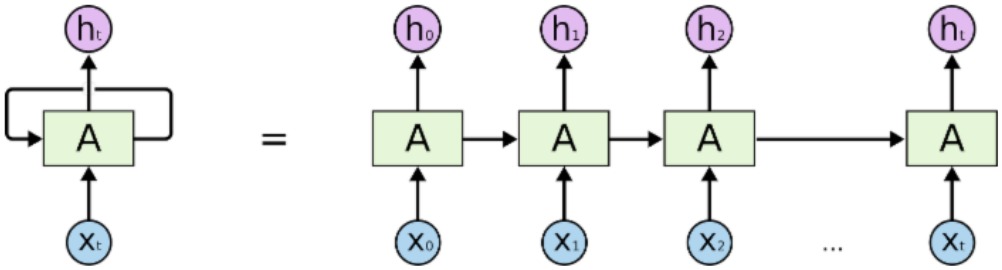


图3-101 RNN及其展开　来源：Christopher Olah

人工神经网络（如多层感知器MLP）对于输入变量（特征）的顺序不敏感，并假设所有的输入变量之间是相互独立的，对于序列数据而言，这样的特点与假设就再不合适了，比如：在N-Gram语言模型中，我们就看到了前N个词对当前词的影响，假设我们在一个嘈杂的环境中对话，如果某个词没有听清楚，我们也许能根据之前的对话内容猜出这个词的含义；其它的时间序列：比如天气、股票价格也都会受到之前状态的影响。

《安娜·卡列尼娜》的首句是“幸福的家庭無不相似，不幸的家庭各有不幸。”，假设这是我们语料库中的一句话，我们用这句话来训练RNN模型。在图3-88所示的RNN中，输入“Start”表示句子开始，希望输出能“幸福”；输入“幸福”，希望能输出“的”；输入“的”，希望能输出“家庭”；输入“家庭”，希望能输出“无”…。第4个词“家庭”出现的概率是考虑到了前面3个词的条件概率（把“START”也计算在内），可表示为：$P(w_4|w_1,w_2,w_3)$。整个句子出现的概率可以表示为：$P(w_1, w_2, w_3, \dots, w_n) = P(w_1)P(w_2|w_1)P(w_3|w_1,w_2)\cdots P(w_n|w_1,w_2,\cdots,w_{n-1})$，n可以是任意自然数，理论上这基本上解决了n-Gram模型的问题。

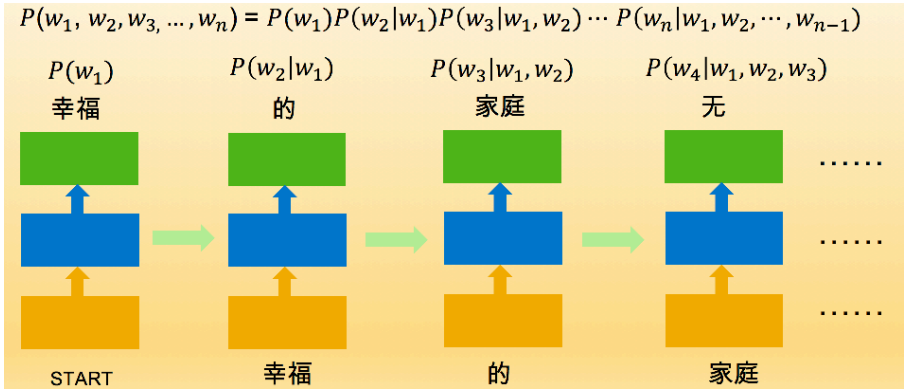


图3-102 基于RNN实现的语言模型

在实践中，有时我们会遇到这样的问题，当我们尝试去预测句子“我在法国长大，…，我能说流利的*法语*”最后的词“*法语*”时，后半句话建议这个词可能是一种语言，但是要搞清楚是什么语言，要找到离这个字可能距离很远（比如在这段话的第一句话）的*法国*才行，RNN虽然理论上能解决这类问题，但实践中，远距离连接的能力往往会丧失。因为RNN也要依靠BP进行模型参数的优化，只不过略微改变了一下形式，变成了***BPTT***（BP Through Time），所以仍会受到梯度下降/爆炸的困扰，对长距离依赖的RNN训练非常困难。1994年，Yoshua Bengio等人就对这个问题进行了深入的研究[i]。

1997年，深度学习泰斗科学家Juergen Schmidhuber等人提出了一种特殊类型的RNN — ***常短时记忆***（Long Short Term Memory，LSTM），因为太领先当时的时代，计算机硬件性能难以支撑这样的算法需求等原因，在当时LSTM并没有引起广泛的关注，直到近年来，他的学生Alex Graves[1]对其

[1] Alex Graves：谷歌DeepMind研究科学家，“神经图灵机”的发明人，师从Juergen Schmidhber获AI博士学位，后在慕尼黑技术大学和多伦多大学进行博士后研究，师从辛顿教授。

进行了改良和推广，才被广泛应用，并取得了巨大的成功，谷歌的研究人员利用LSTM减少了49%的语音识别错误。在深度学习领域，LSTM可以说是与CNN的比肩的最重要的原创性想法之一。

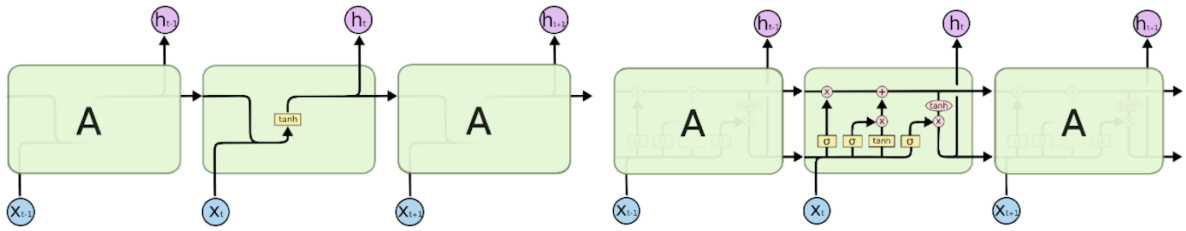


图3-103 标准RNN与LSTM的结构对比 来源：Christopher Olah

标准的RNN中不同时间步（time step）所重复的单元只含有一个非常简单的结构 — 双曲正切（*tanh*）层，LSTM是一种特殊的RNN，总体框架与标准RNN相同，只是对重复的单元进行了替换，LSTM用4个协同工作的结构替换了RNN中的双曲正切函数，改进过的结构使状态可以被记忆，并能被“永远“地传递下去，而不会发生标准RNN因为梯度消失或梯度爆炸导致传递的步数（Steps）受到限制的问题。我们在CNN中提到过残差网络将输入与输出端“短路”的思想与高速公路网络（ Highway Network）很相似，而高速公里网络也是由LSTM之父Juergen Schmidhber所发明的！我们从中可以发现解决梯度消失 / 爆炸问题的共同思路就是以某种方式连接输入和输出端（残差网络是直接短路），这样不论经过多少网络层（ResNet）或时间步（LSTM）的传递，状态都可能被完整的保持下来，最强大算法的核心思想竟然如此简单！

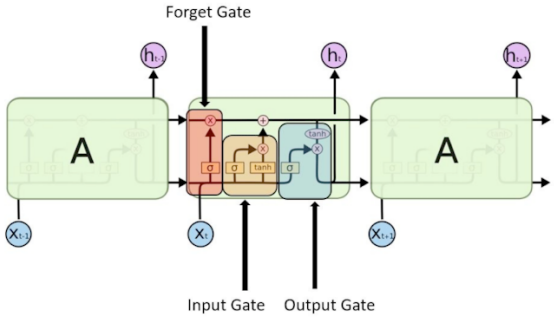
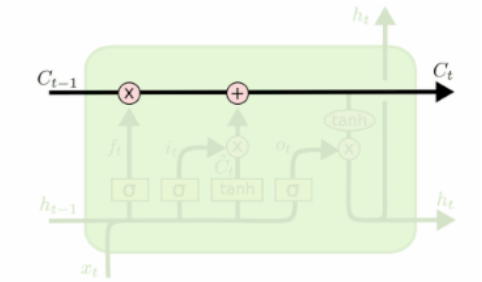
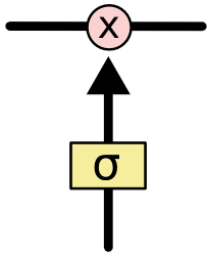


图3-104 LSTM的“三重门”：遗忘门、输入门和输出门

LSTM的关键在于“单元状态”（Cell Status）要能被长时间（经过很多时间步）地传递， 才能解决类似“我在*法国* 长大，... ，我能说流利的*法语*”这样两个词距离遥远的问题。图3-105（ a）中以一根黑色的水平线贯穿运行代表“单元状态” ，连接“单元“的输入和输出端（可对比残差网络的结构），就像是一根传送带， 信息再上面流转，能很容易地长时间保持不变。同时LSTM通过*Sigmoid*门（图b）中以σ 表示）来保护和控制“单元状态” ，我们知道*Sigmoid*函数的输出范围在0到1之间，它定义了有多少比例的信息可以通过， “1”表示“完全通过” ， “0”表示“完全禁止通过” ，可以把*Sigmoid*门想象为一个“阀门”。LSTM有3个这样的“阀门“：**遗忘门**（Forget Gate）、**输入门**（Input Gate）和**输出门**（Output Gate）。





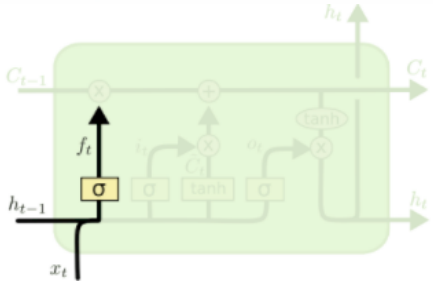


图3-105 LSTM（a）“单元状态”传送带 (b) Sigmoid门 (c) 遗忘门 来源：Christopher Olah

LSTM的第一步是要决定“单元状态”中的哪些信息是要被“遗忘”的（图c）。这个决定由“三重门”中的第1重门 —“遗忘门” 所决定。遗忘门通过*Sigmoid*函数对t 时刻的输入 x_t和（$t-1$）时刻隐层的输出 h_{t-1}进行整合， 得到的f_t的取值范围都在0到1之间[2]，该结果决定了上一个“单元状态” C_{t-1}有哪些信息能保留下来。举例来说：在英语中，单元状态中应该包含当前主语的性别，这样模型才能选择使用正确的人称代词。

下一步是要决定新信息要如何被保存到“单元状态”中，这个功能有两部分组成：首先，双曲正切（Tanh）层创建了一个新的候选值的向量$\tilde{C}_t$[3]，这个新向量将会被加入“单元状态 “；其次，一个被称为 *“输入门”* 的*Sigmoid*层决定了如何$\tilde{C}_t$如何更新单元状态[4]（图d）。然后将这两部的输出结果整合一下，就可以更新“单元状态”了，t 时刻的状态C_t等于（$t-1$）时刻的状态C_{t+1}乘以遗忘门的输出f_t加$\tilde{C}_t$乘以输入门的结果的i_t[5]（图e）。回到上面语言模型的例子中，我们希望将新的主语添加到单元状态中，以替换我们希望忘记的原有主语。注意到更新单元状态的符号是✚号，而不是*Sigmoid*的✖号，如果说✖号决定了信息通过的百分比，类似与“阀门”的话，那么✚就可以类比为“三通” ，在“单元状态”的传送带主线上，汇入了一股经过精心调节的新数据流，更新了“单元状态。

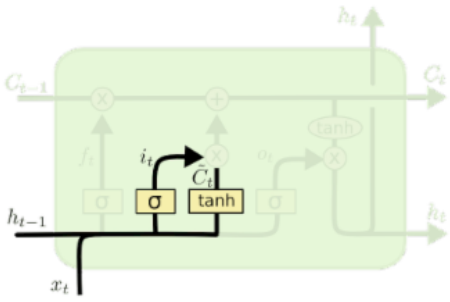
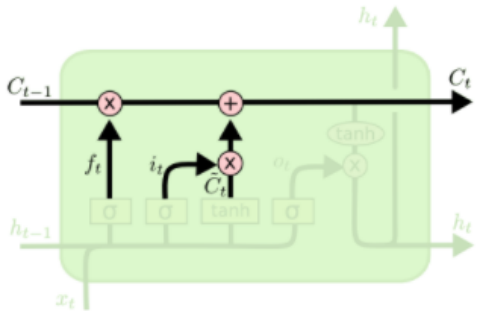


图3-105(d) 输入门



(e)更新“单元状态” 来源：Christopher Olah

[2] $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$, 其中$W_f$是权重，$b_f$为偏置（Bias），下标 f 表示遗忘门（Forget）。
[3] $\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$
[4] $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$
[5] $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$

最后一步，我们要决定“单元”如何输出（图f），基于“单元状态“的输出，同样要经过Sigmoid“阀门”的”过滤“，这个门被称为”*输出门*“，它与前两个相同，都是以x_t和 h_{t-1}作为输入，输出0到1之间的值o_t作为阀门，来控制经过$tanh$挤压的“单元状态”作为隐层h_t 输出，$tanh$函数将值”挤压”到-1和1之间[6]。

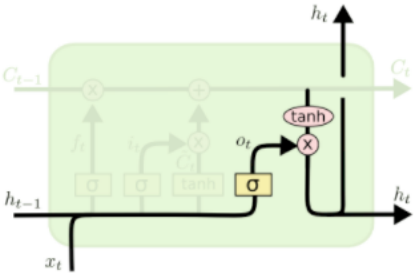


图3-105(f) 输出门

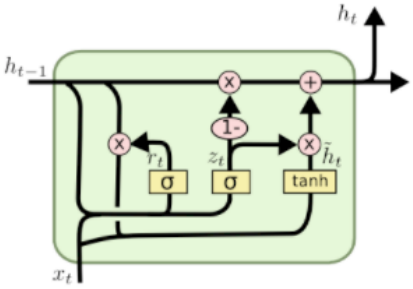
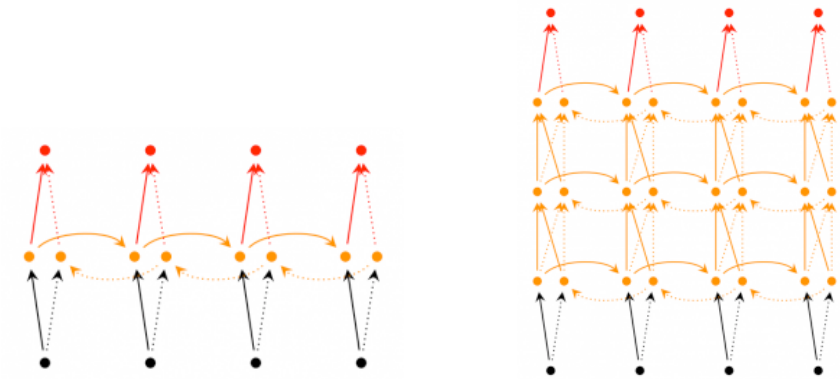


图3-106 GRU 来源：Christopher Olah

Juergen Schmidhber在1997年就提出了标准的LSTM模型，后来有不少学者对其进行了各种微调。其中比较有影响的是2014年由纽约大学Kyunghyun Cho等人提出的“**门循环单元**”（Gated Recurrent Unit, GRU）[ii]，它将遗忘门与输入门合并为“更新门”（Update Gate），也将“单元状态”与“隐层状态”进行了合并和其它一些改变，因为GRU比LSTM少了一个门，结构更简单，也日渐流行。以上LSTM精美的图片及主要内容来源于谷歌大脑的青年才俊Christopher Olah，在此特别感谢他授权本书使用这些内容。

对语言模型而言，有时仅仅看前面的词还不够，不如下面这句话：“我的手机坏了，我打算 ____ 一部新手机。”如果只看横线前面的词，很难判断是买一部新手机还是去修理手机，但是当我们看到“一部新手机”时，横线上的词是“买”的概率就大幅度上升了。因此，我们不仅需要看前面的词，而且可能需要看后面的词，这就需要**双向循环神经网络**（Bidirectional RNNs）了（图3-107左）。

RNN的本质是在时间维度上展开的前馈式神经网络，对于时刻t而言，除了当前输入x_t之外，（t-1）时刻隐层的状态 h_{t-1}也会一起输入“单元”，而h_{t-1}表示从开始到（t-1）时刻所有内容的汇总。双向RNN意味着在玩文字填空游戏时，同时综合考虑空格左右两面的内容。双向RNN的构建非常简单，只需将一个RNN叠加在另一个RNN之上就可以了，输出y_t是基于这两个RNN的隐层状态计算的。标准的RNN只有一个隐层，表达力有限，可以在对双向RNN进行进一步扩展。加入更多的隐层，成为学习能力更强的**深度双向循环神经网络**（Deep Bidiretional RNNs）（图3-107右），当然也需要更多的数据进行训练。



[6] $o_t = \sigma(w_o \cdot [h_{t-1}, x_t] + b_o)$； $h_t = o_t * tanh\,(C_t)$

图3-107 双向RNN与深度双向RNN

谷歌与微软的**机器翻译**（Machine Translation）系统使用的都是**神经翻译机**（Neural Machine Translation，NMT）算法（图3-108），NMT与传统基于统计的翻译模型不同，NMT通过编码器将源语句映射为固定长度的向量，然后解码器基于该向量生成翻译并输出。这种根据输入序列x生成对应的输出序列y的任务被称为**序列到序列**（seq2seq），例如翻译系统一边输入英文序列，一边输出中文序列， 常见的seq2seq任务比如机器翻译和自动问答系统等。

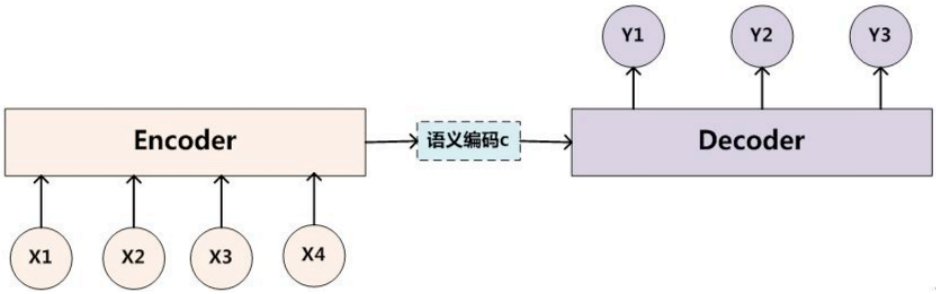


图3-108 神经翻译机

大部分NMT都使用RNN进行编码和解码，图3-109显示的是一个德语翻译成英语的例子：三个德语单词“Echt”、“dicke”和“Kiste” 被依次输入编码器，在终止信号（图中未显示）后，解码器开始生成翻译后的目标语句，解码器会一直生成单词直到最终产生一个的特殊的终止符号。这个模型堪称经典，但是也存在局限性，最大的问题在于编码器与解码器之间的唯一的联系就是语义编码c，即图3-109中的隐层状态h_3，编码器要将整个源语句序列都压缩到一个固定长度的向量之中，这类似于给文章写摘要，无论文章长度是1000字还是1000万字，最终都要被压缩为固定长度的100字摘要，换言之，该固定长度的向量需要捕捉到整个源语句的所有含义。在NLP中，这个向量被称为“**句嵌入**”（Sentense Embedding），即将句子从一个高维度空间中进行降维，然后嵌入到低维度空间（可以参考词向量中的“词嵌入”Word Embedding的概念，对应前面的例子：将1000字压缩为100字）。如果我们将两个语义接近的句子通过PCA或t-SNE[7]等方法降维，并投影到低维度空间，你可能会发现投影后这两个向量间的距离仍然很接近。

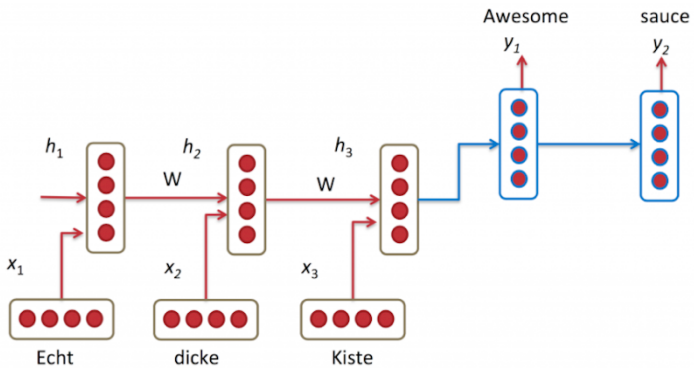


图3-109 神经翻译机

将一个很长的语句压缩到一个低维向量中，这类似于用固定的100字摘要来描述1000个字的文章，再让解码器再基于这个摘要去生成质量很高的翻译，感觉总是还有些问题，因为这是**有损压缩**。另外，在自然语言的翻译中，比如：英语中的第1个词可能与中文中的第1个词高度相关，如果要翻译一个50个词的句子也就是说解码器必需要考虑50步以前的信息，而且50步之前的信息需要被编码至固定

[7] t-SNE(t-distributed stochastic neighbor embedding):辛顿教授等人发明的一种降维的方法，这种非线性降维方法特别适合将高维数据降至2-3维的空间中，并以散点图的形式显进行可视化。

向量之中。我们已知标准的RNN在处理这类长距离依赖时存在梯度消失或爆炸的问题，LSTM进行了很大的改进，但仍未完全解决问题，例如：研究人员发现：把源句子倒过来输入编码器往往能取得更好的效果（双向LSTM），因为这缩短了解码器到编码器对应部分的距离，类似的，把一个语句重复输入2次，也有助于网络更好地“记忆”该语句。把句子倒过来输入是一种“小技巧”，因为只对部分语言之间的翻译有效，例如：德语和法语、英文之间的语序基本类似，这样做是有效的（甚至中文与英文的语序都很接近），但是，某些语言（例如日语）句末词很大程度与翻译后英语的句首词高度相关，颠倒句序会使翻译变的更糟。因此，我们需要一种更彻底的解决方案！

注意力是一种有限的资源，我们不可能同时关注环境中的所有对象，如何才能将不重要的信息过滤掉，什么时间点又要开始注意新的对象？当我们做翻译工作时，我们会特别注意正在翻译的那个单词；当我们在描述周围环境时，我们会把目光移向正在描述的物体上。心理学家已经对人类的视觉注意力机制进行过深入地研究，主要的发现是：在某一时刻，注意力以“高分辨率”模式聚焦于某一特定区域，而对周围区域则采用“低分辨率”模式。机器学习中**注意力**（Attention）模型就是受到这种机制的启发，于2010年左右在玻尔兹曼机的应用中被提出，但直到最近，才在自然语言处理领域的RNN / LSTM应用中流行起来，并被“反哺”回视觉应用中。

在某一时刻，（机器学习）注意力仅关注部分信息子集。为了便于机器学习的进行，我们希望注意力是连续值，所以将模型定义为：关注所有的位置，只是程度不同。基于注意力机制，我们不再试图将一个完整的输入语句编码成一个固定长度的向量。相反，我们在形成输出的每一步都允许解码器「关注」到输入句子的不同部分。更重要的是，我们会让这个模型基于输入的句子以及到目前为止产生的结果去学习应该「关注」什么。所以，在一些高度一致的语言（如英语、德语、法语）中，解码器将可能选择按照顺序进行「关注」。翻译第一个英语单词时，就「关注」第一个德语单词；而在英文到日文（英文与中文的顺序也是比较一致的）的翻译中，注意力就不再按顺序分布了。图3-110（a）中，Y是由解码器输出的翻译结果，X是输入编码器的源序列。编码器是双向RNN / LSTM，但这不重要，可以先忽略逆向，关键在于每次解码器输出y_t依赖于**所有输入状态的权重组合**，而不仅仅是最后的状态。α定义了每个输入状态对于输出的权重，例如，如果$\alpha_{3,2}$的值很大，意味着解码器在生成第3个目标词时，在源语句产生的第2个状态上分配了很多注意力，所有的α之和一般等于1。图3-101(b)显示了权重矩阵α 的可视化，直观的显示了翻译的过程，这个例子是从法语翻译到英语，网络的注意力基本上按顺序聚焦于每个输入状态，但有时在输出时， 比如在将“la Syrie” 翻译为 “Syria” 之时，会同时注意2个词。[iii]

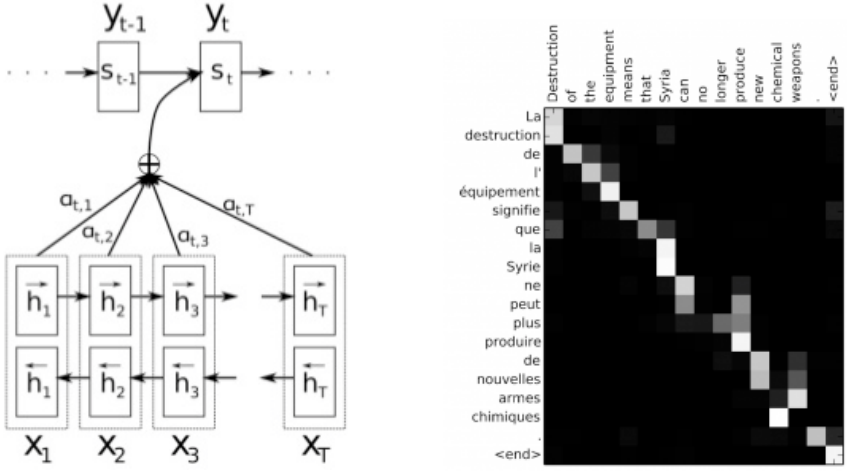


图3-110(a) 使用注意力模型的NMT　　(b)翻译过程中的注意力权重矩阵α 的可视化

除了机器翻译，注意力模型也被应用于图像自动生成文字（与CNN集成）（图3-111）、自动摘要生成和视频目标识别等其它领域。 2016年6月，谷歌发表了一篇关于注意力模型的论文《Attention is all what you need!》[iv]，文章进行了大胆的创新，提出完全舍弃RNN和CNN，从自然语言的特点出发，仅仅使用注意力模型，就能获得非常好的表现，同时还具有计算量小和适合并行等优点，估计未来注意力模型可能会成为一个新的热点，结合其他模型一起创新。

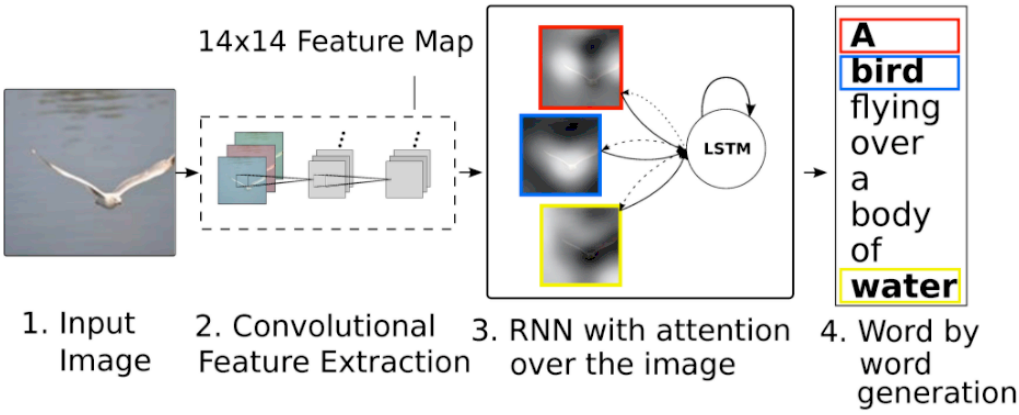


图3-111 基于图像自动生成描述：集成了CNN，RNN和注意力模型　来源：Kelvin Xu等[v]

[i] http://www-dsi.ing.unifi.it/~paolo/ps/tnn-94-gradient.pdf
[ii] https://arxiv.org/pdf/1406.1078v3.pdf
[iii] https://arxiv.org/abs/1409.0473
[iv] https://arxiv.org/pdf/1706.03762.pdf
[v] https://arxiv.org/abs/1502.03044