1 序列模型和注意力机制

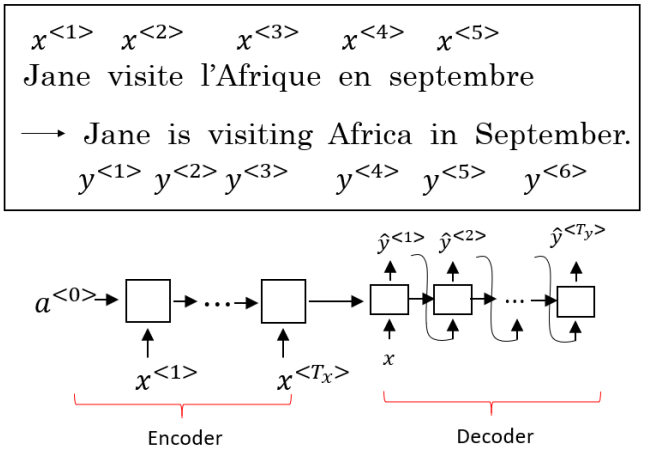
采用循环神经网络能够建立各种各样的**序列模型（Sequence Model）**。加入一些注意力机制，能够使这些序列模型更加强大。

1.1 Seq2Seq 模型

1.1.1 Sequence to Sequence model

2014 年 Cho 等人在论文 Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation (https://arxiv.org/abs/1406.1078) 中首次提出了 Seq2Seq（Sequence-to-Sequence）模型。从机器翻译到语音识别，这种模型能够在各种序列到序列的转换问题中得到应用。

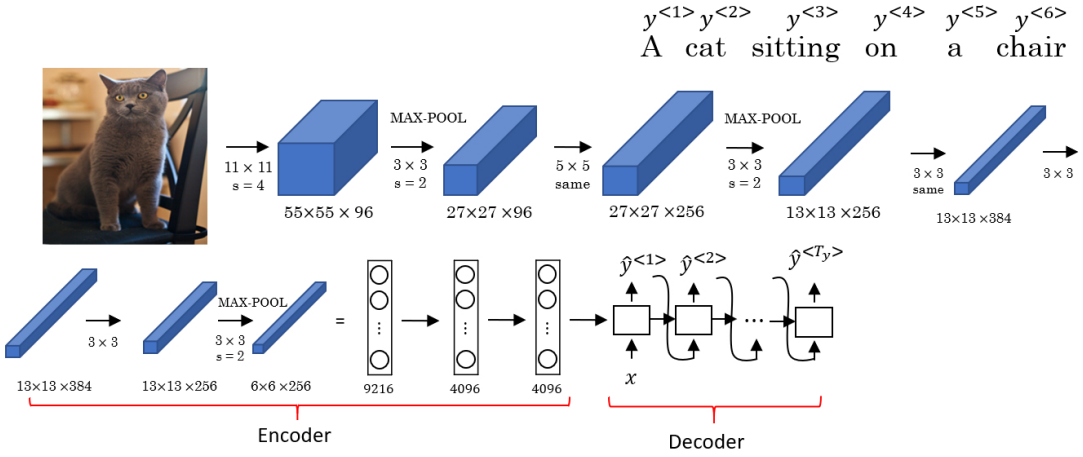
一个 Seq2Seq 模型中可分成**编码器（Encoder）**和**译码器（Decoder）**两部分，它们通常是两个不同的神经网络。如下图是谷歌机器翻译团队的 Sutskever 等人 2014 年在论文 Sequence to Sequence Learning with Neural Networks (https://arxiv.org/pdf/1409.3215.pdf) 中提出的机器翻译模型：



上面法语中的词依次输入作为编码器的 RNN 的时间步中，这个 RNN 可以是 GRU 或 LSTM。将编码器的最后输出作为译码器的输入，译码器也是一个 RNN，训练译码器作出正确的翻译结果。

1.1.2 image to sequence model

这种 Enconder-Decoder 的结构，也可以应用在图像标注（Image Caption）上。2014 年百度研究所的毛俊骅等人在论文 DDeep Captioning with Multimodal Recurrent Neural Networks (m-RNN) (https://arxiv.org/pdf/1412.6632.pdf) 中提出了如下图中的结构：

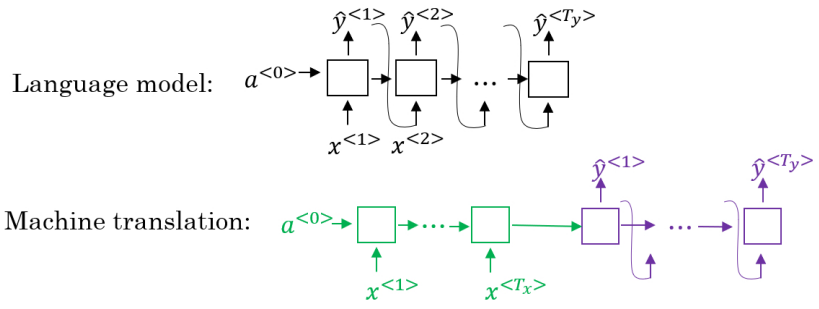


上面将图像输入了一个作为编码器的 AlexNet 结构的 CNN 中，最后的 Softmax 换成一个 RNN 作为译码器，训练网络输出图像的标注结果。

另外两篇论文 Show and Tell: A Neural Image Caption Generator (https://arxiv.org/pdf/1411.4555.pdf) 和 Deep Visual-Semantic Alignments for Generating Image Descriptions (https://arxiv.org/pdf/1412.2306.pdf) 中， 也提到了这种结构。

1.1.3 挑选最可能的句子

机器翻译用到的 Seq2Seq 模型中，译码器所做的工作与前面讲过的语言模型的采样过程类似，只不过在机器翻译中，用编码器的输出代替语言模型中的 0 作为译码器中第一个时间步的输入，如下图所示：



换句话说，机器翻译其实是一个输入为法语的条件下，希望以对应的英语作为输出的语言模型，所以机器翻译的过程也就相当于建立一个**条件语言模型（Conditional Language Model）**的过程。

采用大量的数据训练好一个机器翻译系统后，对于一个相同的句子，由于译码器进行的是随机采样过程，输出的可能会是多种或好或坏的结果，因此需要找到能使条件概率最大化的翻译，即：

$$arg\ max_{y^{\langle 1\rangle},\ldots,y^{\langle T_y\rangle}} P(y^{\langle 1\rangle},\ldots,y^{\langle T_y\rangle}|x) \tag{1}$$

Jane visite l'Afrique en septembre. $P(y^{<1>},\ldots,y^{<T_y>}|x)$

- Jane is visiting Africa in September.
- Jane is going to be visiting Africa in September.
- In September, Jane will visit Africa.
- Her African friend welcomed Jane in September.

$$\underset{y^{<1>},\ldots,y^{<T_y>}}{\arg\max}\ P(y^{<1>},\ldots,y^{<T_y>}|x)$$

对于我们的机器翻译模型来说，如果使用贪心搜索算法，在生成第一个词的分布后，贪心搜索会根据我们的条件语言模型挑选出最有可能输出的第一个词语，然后再挑选出第二个最有可能的输出词语，依次给出所有的输出。

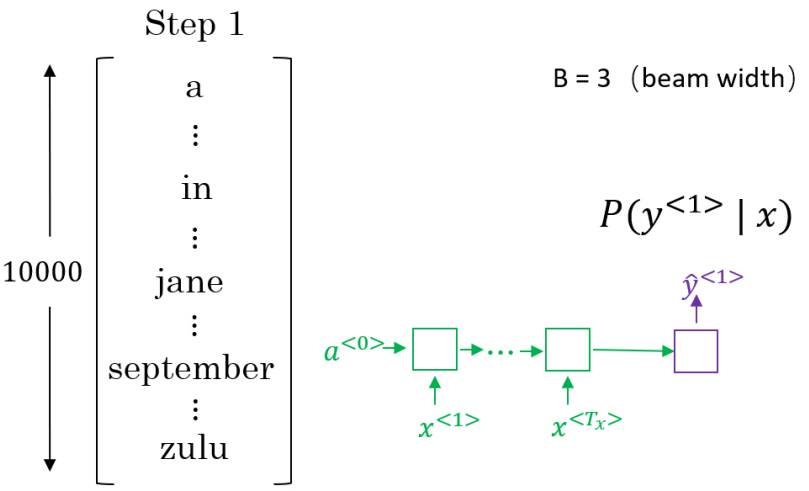
但是对于我们建立的机器翻译模型来说，我们真正需要的是通过模型一次性地挑选出整个输出序列：$y^{\langle 1\rangle},\ldots,y^{\langle T_y\rangle}$，来使得整体的概率最大化。所以对于贪心搜索来说，这种方法对于机器翻译来说是不可行的。

1.2 集束搜索

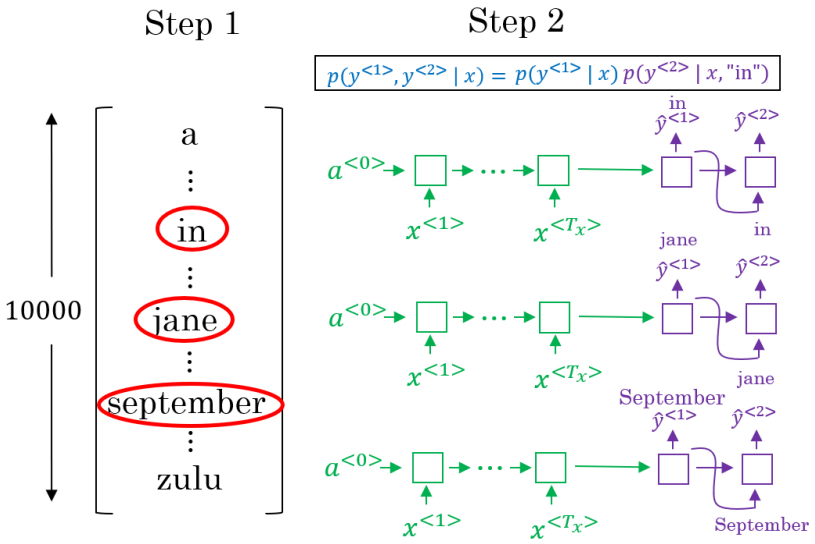
1.2.1 步骤

Seq2Seq 模型中，译码器的输出结果总是在 RNN 中采样后得到，造成模型训练完毕后，得到测试的结果参差不齐，**集束搜索（Beam Search）**算法能很好地解决这个问题。这里还是以机器翻译的例子来说明这种算法。

将集束搜索算法运用到机器翻译系统的第一步，是设定一个**束长（Bean Width）** B，它代表了译码器中每个时间步的预选单词数量。如下图中 $B = 3$，则将第一个时间步中预测出的概率最大的 3 个词作为首词的预选词，同时保存它们的概率值大小 $p(y^{\langle 1 \rangle} \mid x)$:



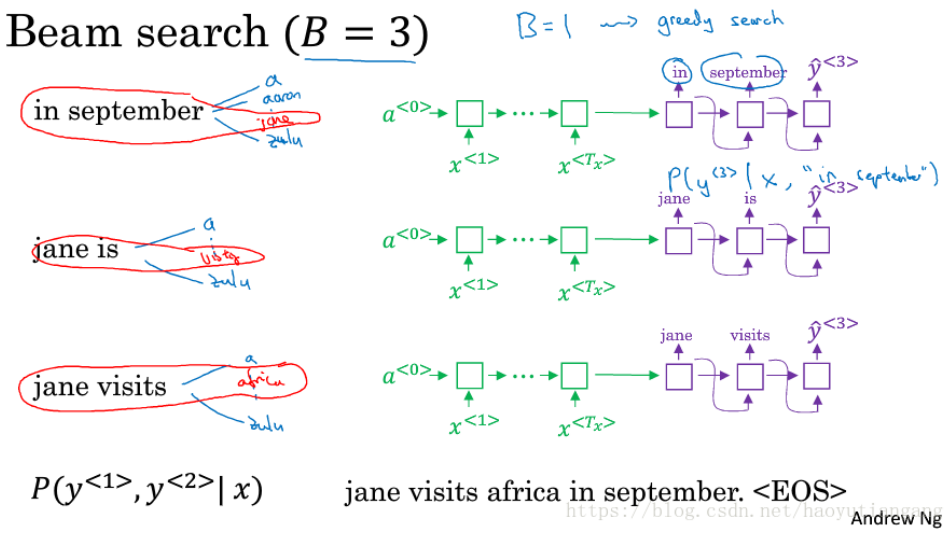
如果第一步得到的三个预选词分别为 “in”、“jane” 和 “September”，如下图所示，则第二步中，分别将三个预选词作为第一个时间步的预测结果 $y^{\langle 1 \rangle}$ 输入第二个时间步，得到预测结果 $\hat{y}^{\langle 2 \rangle}$，也就是条件概率值 $p(\hat{y}^{\langle 2 \rangle} \mid x, y^{\langle 1 \rangle})$:



根据**条件概率**公式，有:

$$p(y^{\langle 1 \rangle}, \hat{y}^{\langle 2 \rangle} \mid x) = p(y^{\langle 1 \rangle} \mid x) p(\hat{y}^{\langle 2 \rangle} \mid x, y^{\langle 1 \rangle}) \tag{2}$$

分别以三个首词预选词作为 $y^{\langle 1 \rangle}$ 进行计算，将得到 30000 个 $p(y^{\langle 1 \rangle}, \hat{y}^{\langle 2 \rangle} \mid x)$。之后还是取其中概率值最大的 $B = 3$ 个，作为对应首词条件下的第二个词的预选词。比如第二个词的预选词分别是 “in” 为首词条件下的 “September”，”jane” 为首词条件下的 “is” 和 “visits”，这样的话首词的预选词就只剩下了 “in” 和 “jane” 而排除了 “September”。后面的过程以此类推，最后将输出一个最优的翻译结果。



可以看到，当 $B = 1$ 时，集束搜索就变为贪心搜索。

1.2.2 优化

总的来说，集束搜索算法所做的工作就是找出符合以下公式的结果：

$$arg\ max \prod_{t=1}^{T_y} p(y^{\langle t \rangle} \mid x, y^{\langle 1 \rangle}, \dots, y^{\langle t-1 \rangle}) \tag{3}$$

然而概率值都是小于 1 的值，多个概率值相乘后的结果的将会是一个极小的浮点值，会造成**数值下溢（Numerical Underflow）**，即累积到最后的结果是一个计算机不能精确表示的极小浮点数。改进的方法**长度标准化（Length Normalization）**，即取上式的 log 值并进行标准化：

$$arg\ max\ \frac{1}{T_y^{\alpha}} \sum_{t=1}^{T_y} log\ p(y^{\langle t \rangle} \mid x, y^{\langle 1 \rangle}, \dots, y^{\langle t-1 \rangle}) \tag{4}$$

其中，T_y 是翻译结果的单词数量，α 是一个需要根据实际情况进行调节的超参数。标准化用于减少对输出长的结果的惩罚（因为翻译结果一般没有长度限制）。

集束宽度的选择

- Beam Width 越大，计算量越大，结果越准确
- Beam Width 越小，计算量越小，结果越不准确
- 一般工作中设置为 3 - 10 即可，研究领域为了追求极致有的设置为 100 - 1000

与 CS 中的精确查找算法：**广度优先查找（Breadth First Search，BFS）**、**深度优先查找（Depth First Search，DFS）**算法不同，集束搜索算法运行的速度虽然很快，但是并不保证能够精确地找到满足 $arg\ max\ p(y \mid x)$ 的结果。

1.2.3 误差分析

集束搜索算法是一种近似搜索算法，也被称为**启发式搜索算法。**而不是一种精确的搜索。

模型分为两个部分：

- RNN 部分：编码网络 + 解码网络
- Beam Search 部分：选取最大的几个值

例如，对于下述两个由人工和算法得到的翻译结果：

$$Human: Jane visits Africa in September. (y^*)$$
$$Algorithm: Jane visited Africa last September. (\hat{y})$$

发现机器翻译的结果 $\hat{y}$ 与专业的人工翻译的结果 y^* 存在较大的差别。要找到错误的根源，首先将翻译没有差别的一部分 “Jane visits Africa” 分别作为译码器中其三个时间步的输入，得到第四个时间步的输出为 “in” 的概率 $p(y^* \mid x)$ 和 “last” 的概率 $p(\hat{y} \mid x)$，比较它们的大小并分析：

- $p(y^* \mid x) > p(\hat{y} \mid x)$的情况：Beam search 算法选择了 $\hat{y}$，但是 y^* 却得到了更高的概率，所以 Beam search 算法出错了；
- $p(y^* \mid x) \leq p(\hat{y} \mid x)$的情况：翻译结果 y^* 相比 $\hat{y}$ 要更好，但是 RNN 模型却预测 $p(y^* \mid x) \leq p(\hat{y} \mid x)$，所以这里是 RNN 模型出现了错误。

建立一个如下图所示的表格，记录对每一个错误的分析，有助于判断错误出现在 RNN 模型还是集束搜索算法中。如果错误出现在集束搜索算法中，可以考虑增大集束宽 B；否则，需要进一步分析，看是需要正则化、更多数据或是尝试一个不同的网络结构。

Human	Algorithm	$P(y^*\|x)$	$P(\hat{y}\|x)$	At fault?
Jane visits Africa in September.	Jane visited Africa last September.	2×10^{-10}	1×10^{-10}	B
				R

1.3 Bleu 指标

BLEU（Bilingual Evaluation Understudy）是一种用来评估机器翻译质量的指标，它早在 2002 年由 Papineni 等人在论文 BLEU: a Method for Automatic Evaluation of Machine Translation (http://www.aclweb.org/anthology/P02-1040.pdf) 中提出。BLEU 的设计思想与评判机器翻译好坏的思想是一致的：机器翻译的结果越接近专业的人工翻译，则评估的分值越高。

最原始的 BLEU 算法很简单：统计机器翻译结果中的每个单词在参考翻译中出现的次数作为分子，机器翻译结果的总词数作为分母。然而这样得到结果很容易出现错误。

French: Le chat est sur le tapis.

Reference 1: The cat is on the mat.

Reference 2: There is a cat on the mat.

MT output: the the the the the the the.

Precision: $\frac{7}{7}$ Modified precision: $\frac{2}{7}$

如上图的例子中，机器翻译得到的结果是 7 个 “the”，分母为 7，每个 “the” 都在参考翻译中有出现，分子为 7，得到翻译精确度为 1.0，这显然是不对的。改进这种算法，将参考翻译中 “the” 出现的最高次数作为分子，机器翻译结果中 “the” 的出现次数作为分母，可得精度为 $\frac{2}{7}$。

上面的方法是一个词一个词进行统计，这种以一个单词为单位的集合统称为 **uni-gram（一元组）**。以 uni-gram 统计得到的精度 p_1 体现了翻译的充分性，也就是逐字逐句地翻译能力。

Example: Reference 1: The cat is on the mat.

Reference 2: There is a cat on the mat.

MT output: The cat the cat on the mat. $\hat{y}$

两个单词为单位的集合则称为 **bi-gram（二元组）**，例如对以上机器翻译结果（$count$）及参考翻译（$count_{clip}$）以二元组统计有：

bi-gram	$count$	$count_{clip}$
the cat	2	1
cat the	1	0
cat on	1	1
on the	1	1
the mat	1	1
Count	6	4

根据上表，可得到机器翻译精度为 $\frac{4}{6}$。

以此类推，以 n 个单词为单位的集合为 **n-gram（多元组）**，采用 n-gram 统计的翻译精度 p_n 的计算公式为：

$$p_n = \frac{\sum_{\text{n-gram}\in\hat{y}} count_{clip}(\text{n-gram})}{\sum_{\text{n-gram}\in\hat{y}} count(\text{n-gram})} \tag{5}$$

以 n-gram 统计得到的 p_n 体现了翻译的流畅度。将 uni-gram 下的 p_1 到 n-gram 下的 p_n 组合起来，对这 N 个值进行**几何加权平均**得到：

$$p_{ave} = exp(\frac{1}{N}\sum_{i=1}^{N} log^{p_n}) \tag{6}$$

此外，注意到采用 n-gram 时，机器翻译的结果在比参考翻译短的情况下，很容易得到较大的精度值。改进的方法是设置一个**最佳匹配长度（Best Match Length）**，机器翻译的结果未达到该最佳匹配长度时，则需要接受简**短惩罚（Brevity Penalty，BP）**：

$$BP = \begin{cases} 1, & (\text{MT\_length} \geq \text{BM\_length}) \\ exp(1 - \frac{\text{MT\_length}}{\text{BM\_length}}), & (\text{MT\_length} < \text{BM\_length}) \end{cases} \tag{7}$$

最后得到 BLEU 指标为：

$$BLEU = BP \times exp(\frac{1}{N}\sum_{i=1}^{N} log^{p_n}) \tag{8}$$

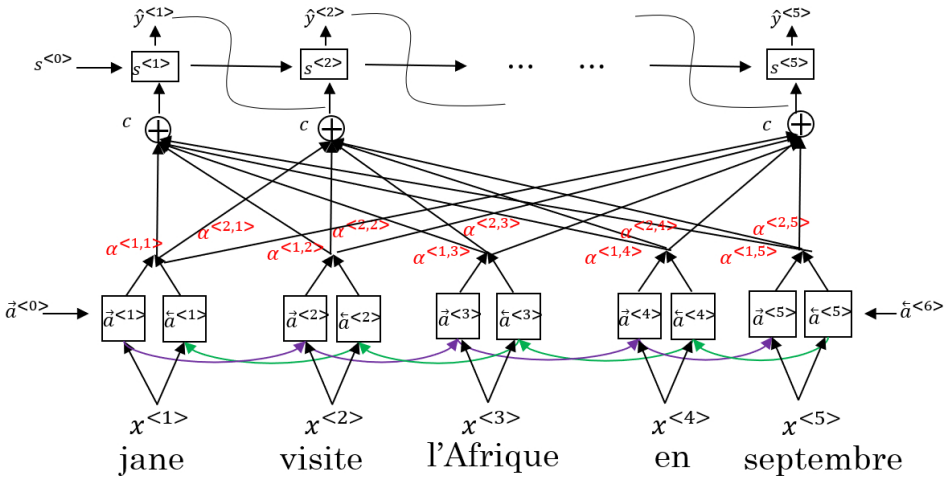
Bleu 得分的贡献是提出了一个表现不错的**单一实数评估指标**，因此加快了整个机器翻译领域以及其他文本生成领域的进程。

1.4 注意力模型

人工翻译一大段文字时，一般都是阅读其中的一小部分后翻译出这一部分，在一小段时间里注意力只能集中在一小段文字上，而很难做到把整段读完后一口气翻译出来。用 Seq2Seq 模型构建的机器翻译系统中，输出结果的 BLEU 评分会随着输入序列长度的增加而下降，其中的道理就和这个差不多。

2014 年 Bahdanau 等人在论文 Neural Machine Translation by Jointly Learning to Align and Translate (https://arxiv.org/pdf/1409.0473.pdf) 中提出了**注意力模型（Attention Model）**。最初只是用这种模型对机器翻译作出改进，之后其思想也在其他领域也得到了广泛应用，并成为深度学习领域最有影响力的思想之一。

注意力模型中，网络的示例结构如下所示：



底层是一个双向循环神经网络，需要处理的序列作为它的输入。该网络中每一个时间步的激活 $a^{\langle t' \rangle}$ 中，都包含前向传播产生的和反向传播产生的激活：

$$a^{\langle t' \rangle} = (\overrightarrow{a}^{\langle t' \rangle}, \overleftarrow{a}^{\langle t' \rangle}) \tag{9}$$

顶层是一个 “多对多” 结构的循环神经网络，第 t 个时间步以该网络中前一个时间步的激活 $s^{\langle t-1 \rangle}$、输出 $y^{\langle t-1 \rangle}$ 以及底层的 BRNN 中多个时间步的激活 c 作为输入。对第 t 个时间步的输入 c 有：

$$c^{\langle t \rangle} = \sum_{t'} \alpha^{\langle t,t' \rangle} a^{\langle t' \rangle} \tag{10}$$

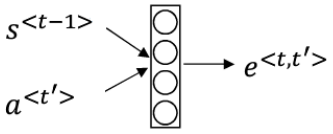
其中的参数 $\alpha^{\langle t,t' \rangle}$ 意味着顶层 RNN 中，第 t 个时间步输出的 $y^{\langle t \rangle}$ 中，把多少 “注意力” 放在了底层 BRNN 的第 t' 个时间步的激活 $a^{\langle t' \rangle}$ 上。它总有：

$$\sum_{t'} \alpha^{\langle t,t' \rangle} = 1 \tag{11}$$

为确保参数 $\alpha^{\langle t,t' \rangle}$ 满足上式，常用 Softmax 单元来计算顶层 RNN 的第 t 个时间步对底层 BRNN 的第 t' 个时间步的激活的 “注意力”：

$$\alpha^{\langle t,t' \rangle} = \frac{exp(e^{\langle t,t' \rangle})}{\sum_{t'=1}^{T_x} exp(e^{\langle t,t' \rangle})} \tag{12}$$

其中的 $e^{\langle t,t' \rangle}$ 由顶层 RNN 的激活 $s^{\langle t-1 \rangle}$ 和底层 BRNN 的激活 $a^{\langle t' \rangle}$ 一起输入一个隐藏层中得到的，因为 $e^{\langle t,t' \rangle}$ 也就是 $\alpha^{\langle t,t' \rangle}$ 的值明显与 $s^{\langle t \rangle}$、$a^{\langle t' \rangle}$ 有关，由于 $s^{\langle t \rangle}$ 此时还是未知量，则取上一层的激活 $s^{\langle t-1 \rangle}$。在无法获知 $s^{\langle t-1 \rangle}$、$a^{\langle t' \rangle}$ 与 $e^{\langle t,t' \rangle}$ 之间的关系下，那就用一个神经网络来进行学习，如下图所示：

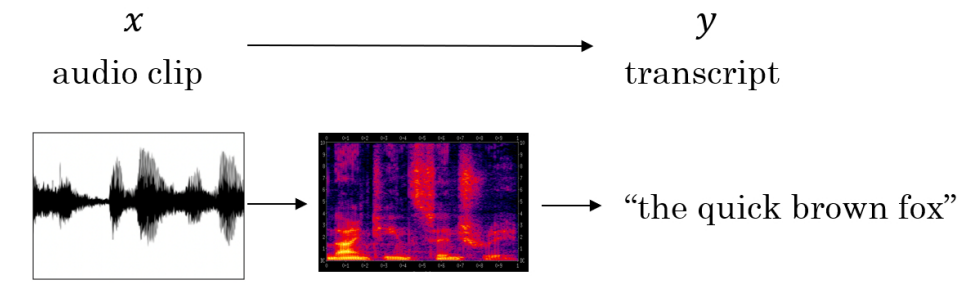


要注意的是，该模型的运算成本将达到 N^2。此外，在 2015 年 Xu 等人发表的论文 Show, Attend and Tell: Neural Image Caption Generation with Visual Attention (https://arxiv.org/pdf/1502.03044.pdf) 中，这种模型也被应用到了图像标注中。

1.5 应用

1.5.1 语音识别

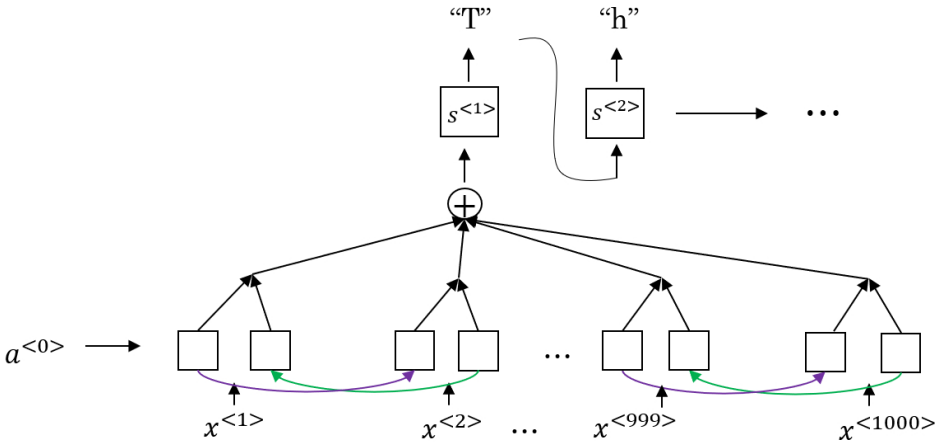
在语音识别中，要做的是将输入的一段语音 x 转换为一段文字副本作为输出。



音频数据的常见预处理步骤是运行音频片段来生成一个声谱图，并将其作为特征。以前的语音识别系统通过语言学家人工设计的**音素（Phonemes）**来构建，音素指的是一种语言中能区别两个词的最小语音单位。现在的端到端系统中，用深度学习就可以实现输入音频，直接输出文本。

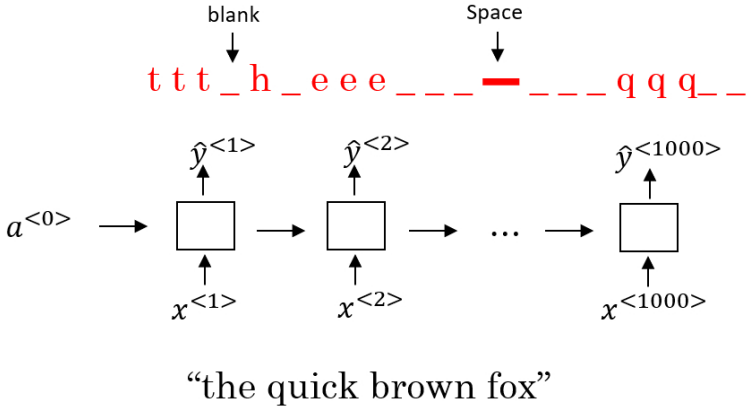
采用深度学习方法训练语音识别系统的前提条件是拥有足够庞大的训练数据集。在学术界的研究中，3000 小时的长度被认为是训练一个语音识别系统时，需要的较为合理的音频数据大小。而训练商用级别的语音识别系统，需要超过一万小时甚至十万小时以上的音频数据。

语音识别系统可以采用注意力模型来构建：



2006 年 Graves 等人在论文 Connectionist Temporal Classification: Labeling unsegmented sequence data with recurrent neural networks (http://people.idsia.ch/~santiago/papers/icml2006.pdf) 中提出了一种名为 **CTC（Connectionist Temporal Classification）**损失函数计算方法，给语音识别系统的训练过程带来很大帮助。

由于输入的是音频数据，采用 RNN 建立的语音识别系统中将包含多个时间步，且整个网络中输出的数量往往是小于输入的数量的，也就是说不是每一个时间步都有对于的输出。而 CTC 的主要优点，是可对没有对齐的数据进行自动对齐。



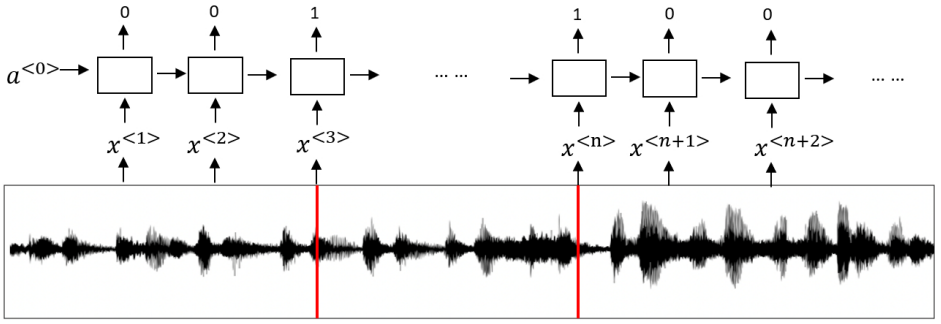
如上图中，以一句意为图中下面的句子，长度为 10s 频率为 100Hz 的语音作为输入，则这段语音序列可分为 1000 个部分，分别输入 RNN 的时间步中，而 RNN 可能不会将 1000 个作为输出。

CTC 损失计算方法允许 RNN 输出一个如图中所示的结果，允许以 “空白（Blank）” 作为输出的同时，也会识别出词之间存在的 “空格（Space）” 标记，CTC 还将把未被 “空白” 分隔的重复字符折叠起来。

1.5.2 触发词检测

触发词检测（Trigger Word Detection）现在已经被应用在各种语音助手以及智能音箱上。例如在 Windows 10 上能够设置微软小娜用指令 “你好，小娜” 进行唤醒，安卓手机上的 Google Assistant 则可以通过 “OK，Google” 唤醒。

想要训练一个触发词检测系统，同样需要有大量的标记好的训练数据。使用 RNN 训练语音识别系统实现触发词词检测的功能时，可以进行如下图所示的工作：



在以训练的语音数据中输入 RNN 中，将触发词对应的序列的标签设置为 “1”，而将其他的标签设置为 “0”，以此训练模型对触发词的检测。

上面方法的缺点就是 0、1 标签的不均衡。一种简单的方法就是在触发字后的多个目标标签都标记为 1，在一定程度上可以提高系统的精确度。