Sequence to Sequence Model - Aug 9'th, 2018

大纲

- seq2seq Model
  - Basic Model
  - Beam Search(集束搜索)
  - Attention Model(注意力模型)
- 对语音数据建模

Sequence to sequence model

Paper: Sutskever et al., 2014. Sequence to sequence learning with neural networks

Paper: Cho et al., 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation

$x^{<1>}$ $x^{<2>}$ $x^{<3>}$ $x^{<4>}$ $x^{<5>}$
Jane visite l'Afrique en septembre
$\longrightarrow$ Jane is visiting Africa in September.
$y^{<1>}$ $y^{<2>}$ $y^{<3>}$ $y^{<4>}$ $y^{<5>}$ $y^{<6>}$

图 1：机器翻译例子（法语 $\longrightarrow$ 英语）

- Many-to-Many
- $T_x \neq T_y$
- 基本的 RNNs Model 架构

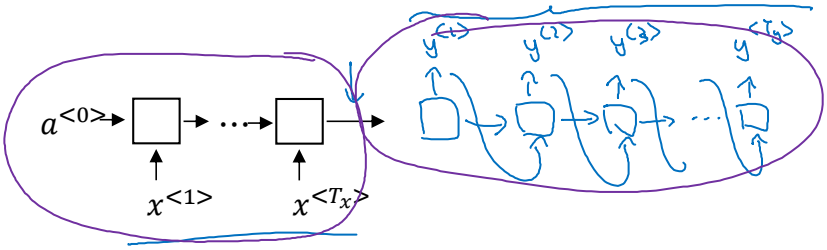

图 2：机器翻译的模型架构图

Image captioning $\longrightarrow$ 图像标识

Reference Paper:

- Mao et. al., 2014. Deep captioning with multimodal recurrent neural networks
- Vinyals et. al., 2014. Show and tell: Neural image caption generator
- Karpathy and Li, 2015. Deep visual-semantic alignments for generating image descriptions

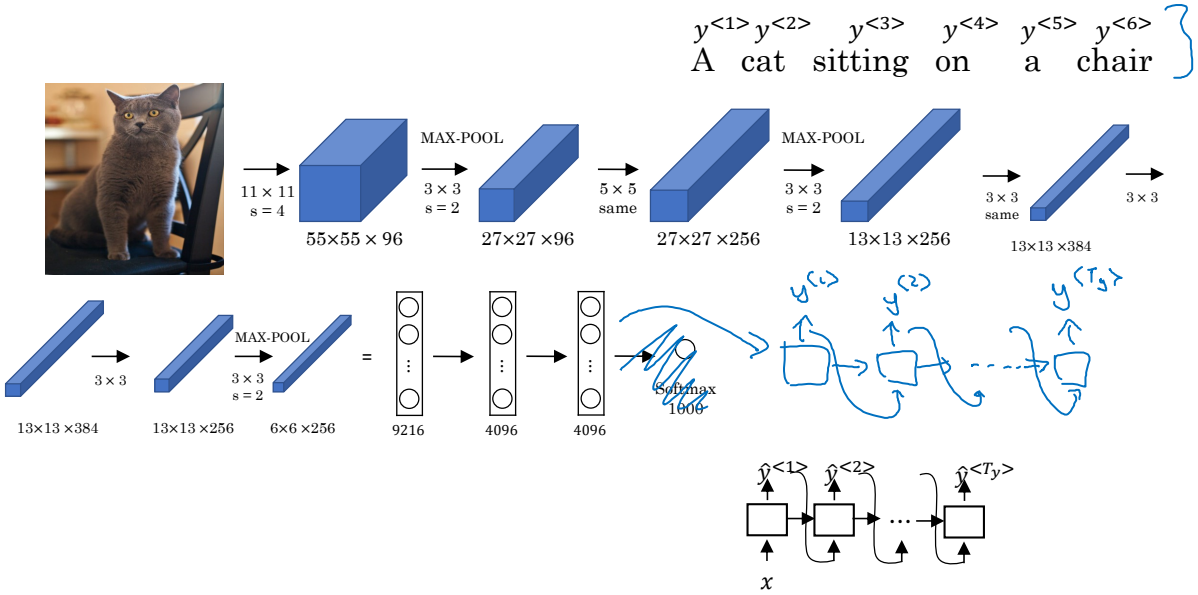

图 3：图像标识的模型结构图（编码器-解码器网络）

Note：seq2seq 模型与传统的语言模型（Language Model or Sequence Generation）最大不同：**输出一个我们自己想要的结果．不是很随意的结果．**

Machine translation vs. Language Model

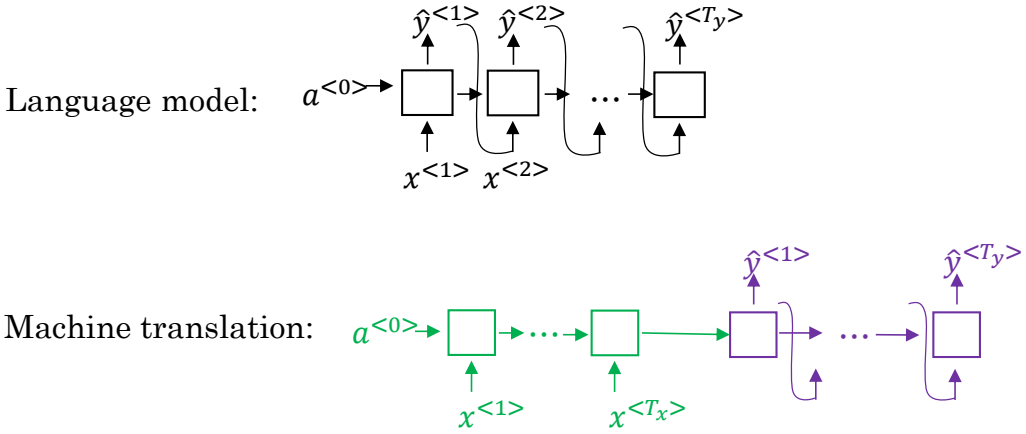

图 4：机器翻译 vs. 语言模型

- 机器翻译 $\longrightarrow$ 有条件的语言模型
- 机器翻译模型的输入不总是全零向量，而是有一个编码网络作为输入.

Finding the most likely translation

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.

图 5：Example ⟶ 找出最合适的翻译

优化目标：

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

Note：如何优化：Greedy Search 和 Beam Search.

Greedy search

greedy search（贪婪搜索机制）是计算机科学的一种算法, 就是第一个字挑的是可能性最大的单词基于你的有条件的语言模型. 进入机器翻译模型, 选择了第一个单词之后, 然后挑出可能性最高的第二个单词, 然后选出可能性最高的第三个单词. 这种算法称为 greedy search（贪婪搜索机制）

事实证明, 贪婪搜索机制, 就是选出最佳的第一个单词在选择后, 试着选出最好的第二个单词, 然后试着选出最好的第三个单词, 这种方法真的行不通.

举例：让我们考虑以下两个翻译. 第一个是更好的翻译, 所以希望在我们的机器翻译模型中, 在给定 x 时更大时 $_1(y)$ 是第一个句子的概率这是一个较好的, 更简洁的法语翻译. 第二个翻译也不是很差, 它只是更啰嗦, 有不必要的词. 但是, 如果算法选择了“jane is”作为前两个字, 因为“going”是一个更常见的英语单词, 对于特定的法语输入，“jane is going,”的可能性实际上概率是高于“Jane is visiting”在这个法语句子中. 所以, 如果你只是在前三个单词最大的概率基础上, 去选择第三个单词，最后很有可能会选择第二个翻译. 但这最终会得出一个不太理想的句子.

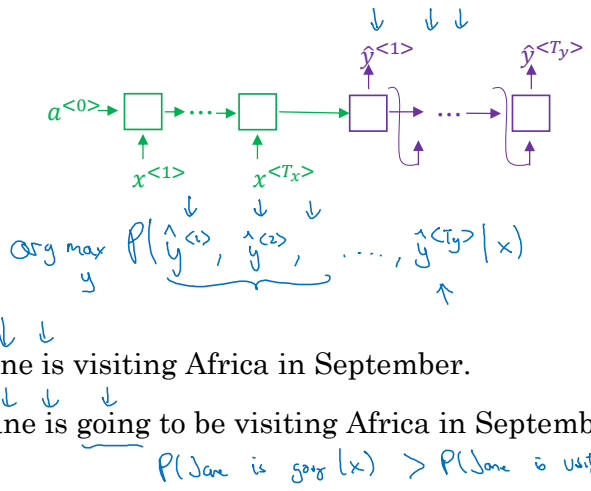

图 6：Why Not a Greedy Search

Note: 每次只选一个单词并不总是最好的

Note: 英语句子中的单词组合总数是指数级别的

如果你只有 1 万字的字典, 然后你要进行长达十个单词的翻译, 然后有 10000^{10} 次方这些可能的句子, 是十个单词. 从词汇量中提取单词, 字典的大小为 10000 词. 所以, 句子空间的可能性是巨大的, 对所有可能评分, 是不可能的这就是为什么最常见的事情是使用一个近似搜索出来. 而且, 近似搜索算法所做的, 是它会尝试, 但不会总是成功, 它会选择使条件概率最大化的句子 y . 而且, 即使它不保证找出 y , 使概率最大化, 但它通常表现不错.

Beam Search

比如法语翻译为英语的机器翻译例子中，给定法语输入句 x 相对于贪心法搜索会只找到一个最可能的单词然后继续下一个，集束搜索会考虑多个可能的选择. 集束搜索有一个参数 B，代表**集束宽度**，在这里我们把它设为 3. 表示集束搜索每次会不仅仅考虑 1 个，而是 3 个可能的选择.

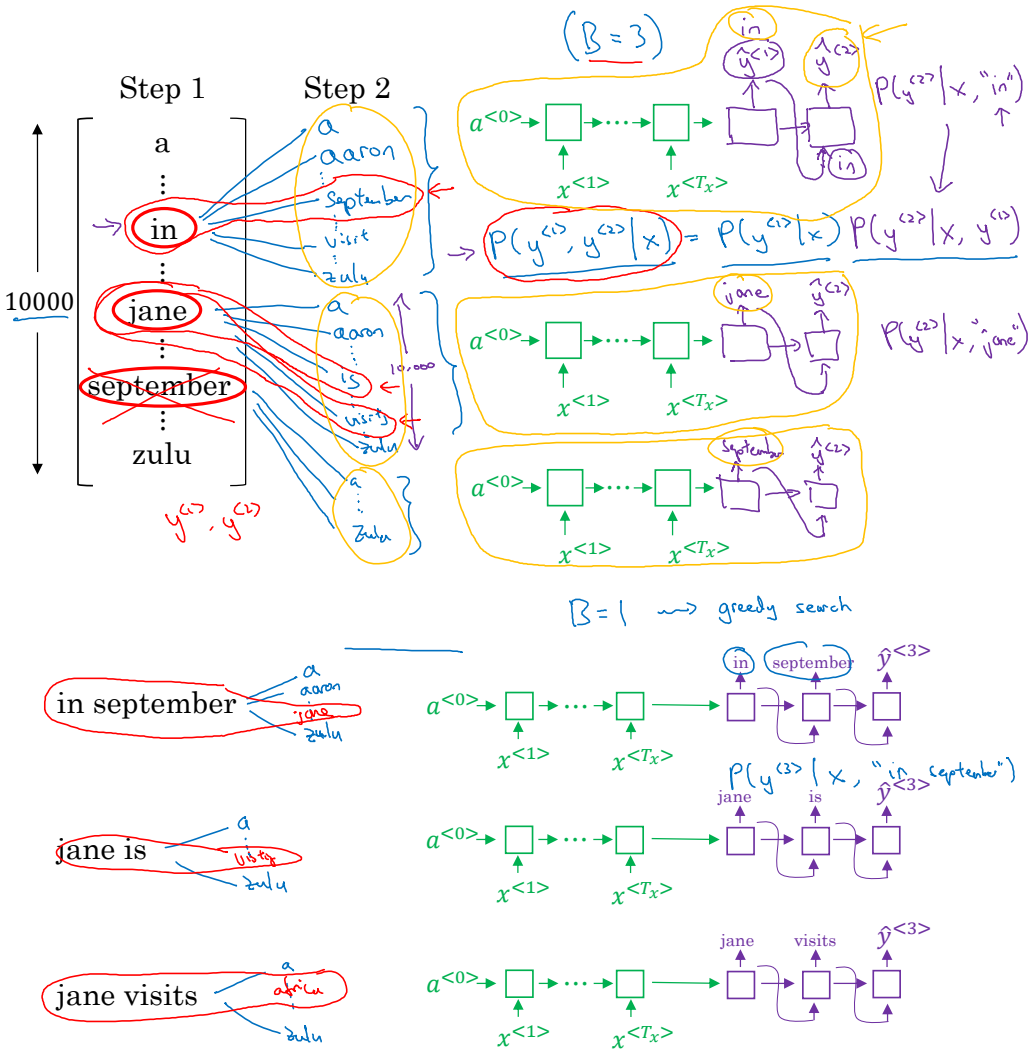

图 7：Beam Search Algorithm.

$$\mathbb{P}(y^{<1>},...,y^{<T_y>}|x) = \mathbb{P}(y^{<1>}|x)\mathbb{P}(y^{<2>}|x,y^{<1>})\cdot\cdots\cdot\mathbb{P}(y^{<T_y}|x,y^{<1>},...) \tag{2}$$

改进 Beam Search

- Length normalization

$$\arg\max_y \prod_{t=1}^{T_y} \mathbb{P}(y^{<t>}|x, y^{<1>}, ..., y^{<t-1>}) \tag{3}$$

$$\arg\max_y \sum_{t=1}^{T_y} \log \mathbb{P}(y^{<t>}|x, y^{<1>}, ..., y^{<t-1>}) \tag{4}$$

$$\frac{1}{T_y^\alpha} \arg\max_y \sum_{t=1}^{T_y} \log \mathbb{P}(y^{<t>}|x, y^{<1>}, ..., y^{<t-1>}) \quad (0 \le \alpha \le 1) \tag{5}$$

- Beam width B
  - Large B: better result, slower
  - Small B: worse result, faster
  - Usually, select $1 \to 3 \to 10$ as B, But some researchers maybe select $1000 \to 3000$ as B, Beacurse attach more result and select the best as experience result of thesis.

Note: Unlike exact search algorithms like BFS (Breadth First Search) or DFS (Depth First Search), Beam Search runs faster but is not guaranteed to find exact maximum for $\max_y \mathbb{P}(y|x)$.

Beam Search 的误差分析

Jane visite l'Afrique en septembre.

Human: Jane visits Africa in September. (y^*)

Algorithm: Jane visited Africa last September. $(\hat{y})$ ⇐

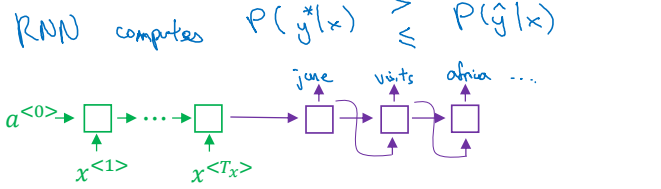

图 8：Example

- 举例而言，对一个句子，“Jane visits Africa in September”你插入“jane”，“visits”，“africa”. 同样, 我现在忽略了大小写的区别, 对, 继续. 计算 $\mathbb{P}(y|x)$. 所以事实证明, 对你现在而言，最重要的事是用这个模型去计算 $\mathbb{P}(y^*|x)$ 和计算 $\mathbb{P}(\hat{y}|x)$. 然后看这两者中哪一个更大. 因此, 左端可能比右端大. $\mathbb{P}(y^*|x) < \mathbb{P}(\hat{y}|x)$ 也是有可能的，实际上小于或等于，对吗？根据这两种情况中的哪一个是正确的, 你可以更清楚地把这个特殊的错误, 这个特别差的翻译归因于 RNN 或者定向搜索算法中的一个有更多的问题. 因此, 让我们理一理这背后的逻辑. 以下是之前幻灯片中的两个句子. 记住, 我们要计算 $\mathbb{P}(y^*|x)$ 和 $\mathbb{P}(\hat{y}|x)$, 看其中哪个更大. 有以下两种情况：
  1. $\mathbb{P}(y^*|x) > \mathbb{P}(\hat{y}|x)$，但是定向搜索算法选择了 $\hat{y}$，定向搜索其实并没有给你最大化 $\mathbb{P}(y|x)$ 的那个 y 的值，因为定向搜索的一个任务就是找到使得函数值非常大的 y. 但它选择了 $\hat{y}$，而 y^* 实际上得到了更高的值. 所以在这个情况下，你可以总结出定向搜索有问题.
  2. $\mathbb{P}(y^*|x) \le \mathbb{P}(\hat{y}|x)$，相比于 $\hat{y}$, y^* 是一个更好的翻译. 但根据 RNN, $\mathbb{P}(y^*) < \mathbb{P}(\hat{y})$, 所以说，相比于 $\hat{y}$，y^* 是一个不太可能的输出结果. 因此, 在这种情况下, 似乎 RNN 模型是有问题的, 而且也许值得花更多的时间在 RNN 上.

注意力机制

- 机器翻译模型（Encoder-Decoder）模型的改进，是深度学习的重要思想之一.
- Paper: Bahdanau et. al., 2014. Neural machine translation by jointly learning to align and translate

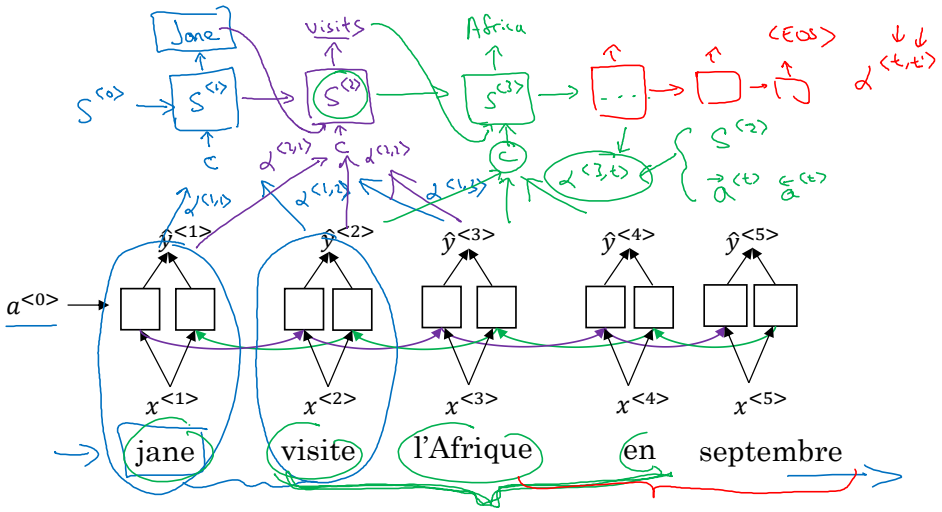

图 9：注意力机制

- 机器翻译模型（Encoder-Decoder）模型的改进，是深度学习的重要思想之一.
- Paper: Bahdanau et. al., 2014. Neural machine translation by jointly learning to align and translate

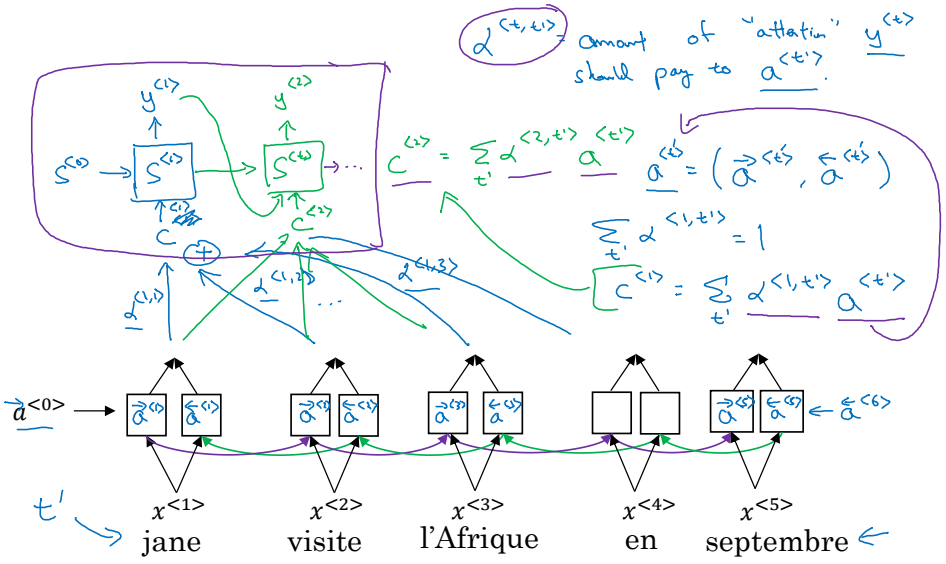

图 10：Attention Model

计算关注度 $\alpha^{<t,t'>}$

Paper: Bahdanau et. al., 2014. Neural machine translation by jointly learning to align and translate

Paper: Xu et. al., 2015. Show, attend and tell: Neural image caption generation with visual attention

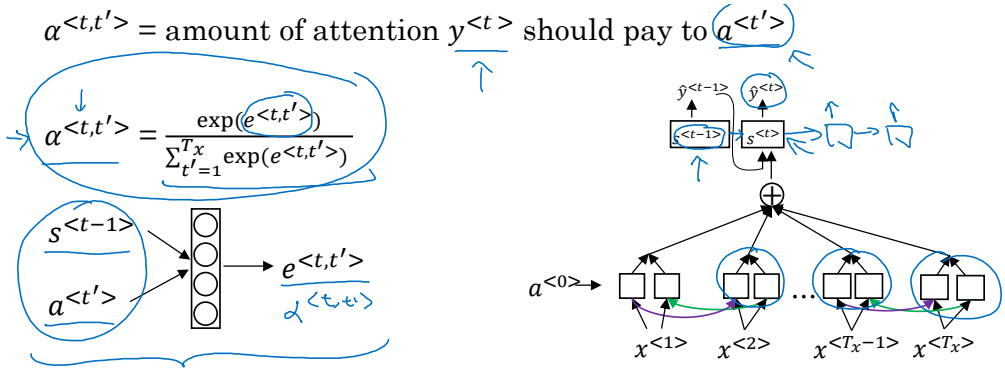

图 11：计算关注度 $\alpha^{<t,t'>}$

Attention examples

July 20th 1969 ⟶ 1969 − 07 − 20

23 April, 1564 ⟶ 1564 − 04 − 23

Visualization of $\alpha^{<t,t'>}$:

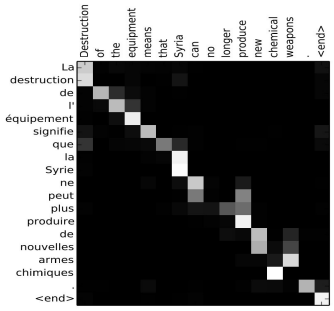

图 12：Attention examples

注意力机制介绍

动机

以英语 - 法语翻译为例，给定一对英语输入序列“They”、“are”、“watching”、“.”和法语输出序列“Ils”、“regardent”、“.”。解码器可以在输出序列的时间步 1 使用更集中编码了“They”、“are”信息的背景变量来生成“Ils”，在时间步 2 使用更集中编码了“watching”信息的背景变量来生成“regardent”，在时间步 3 使用更集中编码了“.”信息的背景变量来生成“.”。这看上去就像是在解码器的每一时间步对输入序列中不同时间步编码的信息分配不同的注意力。这也是注意力机制的由来。它最早由 Bahanau 等提出。

设计

在时间步 t，设解码器的背景变量为 $c_{t'}$，输出 $y_{t'}$ 的特征向量为 $\boldsymbol{y}_{t'}$。和输入的特征向量一样，这里每个输出的特征向量也是模型参数。解码器在时间步 t 的隐藏状态

$$\boldsymbol{s}_{t'} = g(\boldsymbol{y}_{t'-1}, \boldsymbol{c}_{t'}, \boldsymbol{s}_{t'-1}).$$

令编码器在时间步 t 的隐藏状态为 $\boldsymbol{h}_t$，且总时间步数为 T。解码器在时间步 t 的背景变量为

$$\boldsymbol{c}_{t'} = \sum_{t=1}^{T} \alpha_{t't}\boldsymbol{h}_t,$$

其中 $\alpha_{t't}$ 是权值。也就是说，给定解码器的当前时间步 t，我们需要对编码器中不同时间步 t 的隐藏状态求加权平均。这里的权值也称注意力权重。它的计算公式是

$$\alpha_{t't} = \frac{\exp(e_{t't})}{\sum_{k=1}^{T} \exp(e_{t'k})},$$

其中 $e_{t't} \in \mathbb{R}$ 的计算为

$$e_{t't} = a(\boldsymbol{s}_{t'-1}, \boldsymbol{h}_t).$$

上式中的函数 a 有多种设计方法。Bahanau 等使用了多层感知机：

$$e_{t't} = \boldsymbol{v}^\top \tanh(\boldsymbol{W}_s \boldsymbol{s}_{t'-1} + \boldsymbol{W}_h \boldsymbol{h}_t),$$

其中 v、$\boldsymbol{W_s}$、$\boldsymbol{W_h}$ 以及编码器与解码器中的各个权重和偏差都是模型参数。

Bahanau 等在编码器和解码器中分别使用了门控循环单元。在解码器中，我们需要对门控循环单元的设计稍作修改。解码器在 t' 时间步的隐藏状态为

$$\boldsymbol{s}_{t'} = \boldsymbol{z}_{t'} \odot \boldsymbol{s}_{t'-1} + (1 - \boldsymbol{z}_{t'}) \odot \tilde{\boldsymbol{s}}_{t'},$$

其中的重置门、更新门和候选隐含状态分别为

$$\begin{aligned}
\boldsymbol{r}_{t'} &= \sigma(\boldsymbol{W}_{yr} \boldsymbol{y}_{t'-1} + \boldsymbol{W}_{sr} \boldsymbol{s}_{t'-1} + \boldsymbol{W}_{cr} \boldsymbol{c}_{t'} + \boldsymbol{b}_r),\\
\boldsymbol{z}_{t'} &= \sigma(\boldsymbol{W}_{yz} \boldsymbol{y}_{t'-1} + \boldsymbol{W}_{sz} \boldsymbol{s}_{t'-1} + \boldsymbol{W}_{cz} \boldsymbol{c}_{t'} + \boldsymbol{b}_z),\\
\tilde{\boldsymbol{s}}_{t'} &= \tanh(\boldsymbol{W}_{ys} \boldsymbol{y}_{t'-1} + \boldsymbol{W}_{ss}(\boldsymbol{s}_{t'-1} \odot \boldsymbol{r}_{t'}) + \boldsymbol{W}_{cs} \boldsymbol{c}_{t'} + \boldsymbol{b}_s).
\end{aligned}$$