Attention 机制总结

1. Neural Machine Translation by Jointly Learning to Align and Translate

本文最重要的贡献是提出了 attention 机制。

提出背景

神经机器翻译（neural machine translation）大都是 encoder-decoder 模型，encoder 把输入序列转化为固定长度的向量，decoder 把该向量转化为翻译结果。The whole encoder-encoder system is jointly trained to maximize the probability of a correct translation given a source sentence.

该方法存在的问题为，需要把句子的所有信息压缩在固定长度的向量中，这使得模型无法处理长句子。

为了解决这个问题，文章对 encoder-decoder 模型进行了改进，即 align 和 translate 同时进行。每当模型生成新的翻译词时，它在原句那些最有可能包含有关信息的位置上进行搜索。这个方法最重要的特点是，它没有尝试将原句的所有部分 encode 到固定长度的向量，而是把原句 encode 到一序列向量，然后在 decode 的时候灵活选用这个序列的子集。

实验表明，该模型比 basic encoder-decoder 方法效果要好很多，且对长句子提升效果更明显。

RNN Encoder-Decoder

文章首先介绍了基础的框架：RNN Encoder-Decoder，在该框架中，encoder 将输入序列（一组向量）$\mathbf{x} = (x_1, \cdots, x_{T_x})$ 转化为向量 c，最常用的方法是 RNN：

$$h_t = f(x_t, h_{t-1})$$

$$c = q(\{h_1, \cdots, h_{T_x}\})$$

其中，$h_t \in \mathbb{R}^n$ 为时刻 t 的隐藏状态，c 是从隐藏状态序列得到的向量，f 和 q 为非线性函数。比如，Sutskever (2014) 文章中，f 为 LSTM，$q(\{h_1, \cdots, h_{T_x}\}) = h_T$。

给定上下文向量 c 和预测好的序列 $\{y_1, \cdots, y_{t'-1}\}$，训练好的 decoder 就可以预测下一个词 $y_{t'}$ 的概率。最终结果 $\mathbf{y}$ 的概率为：

$$p(\mathbf{y}) = \prod_{t=1}^{T} p(y_t | \{y_1, \cdots, y_{t-1}\}, c),$$

其中，$\mathbf{y} = (y_1, \cdots, y_{T_y})$。使用 RNN，每个条件概率可表示为：

$$p(y_t | \{y_1, \cdots, y_{t-1}\}, c) = g(y_{t-1}, s_t, c)$$

其中，g 为非线性函数，s_t 是 RNN 的隐藏状态。

Learning to Align and Translate

文章对 neural machine translation 提出了一种新的框架。

Decoder

新框架中，条件概率表示为：

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

其中，s_i 是 RNN 时刻 i 的隐藏状态，计算公式为：

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

需要注意的是，和上面的 RNN Encoder-Decoder 不同的是，对每个 y_i，上下文向量 c_i 是不同的。

上下文向量 c_i 依赖于序列 $(h_1,\cdots,h_{T_x})$，该序列由 encoder 对输入句子映射而来，每个 h_i 包含了整个句子对第 i 个词周围部分的关注信息，计算方法在下一节。

上下文向量 c_i 为这些 h_i 的加权值：

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

每个 h_j 的权重 α_{ij} 为：

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

其中，$e_{ij} = a(s_{i-1},h_j)$ 是 *alignment model*，用来对位置 j 周围的输入和位置 i 的输出匹配程度打分。alignment model a 为前向神经网络，其参数和模型其它部分一起训练。

以上就是 decoder 的注意力机制。通过注意力机制，encoder 不再需要对源句所有信息编码为固定长度向量。

Encoder: Bidirectional RNN for Annotating Sequences

encoder 部分使用双向 RNN，前后 RNN $\overrightarrow{f}$ 顺序读输入序列（从 x_1 到 x_{T_x}），并计算前向隐藏状态序列 $(\overrightarrow{h}_1,\cdots,\overrightarrow{h}_{T_x})$。后向 RNN $\overleftarrow{f}$ 反向读输入序列（从 x_{T_x} 到 x_1），得到后向隐藏状态序列 $(\overleftarrow{h_1},\cdots,\overleftarrow{h_{T_x}})$。对每个词 x_j，将 $\overrightarrow{h}_j$ 和 $\overleftarrow{h_j}$ 拼接起来，得到 h_j，即 $h_j = \begin{bmatrix}\overrightarrow{h}_j^T, \overleftarrow{h_j}^T\end{bmatrix}^T$。

以上就是注意力机制的思想，模型的整体框架图为：

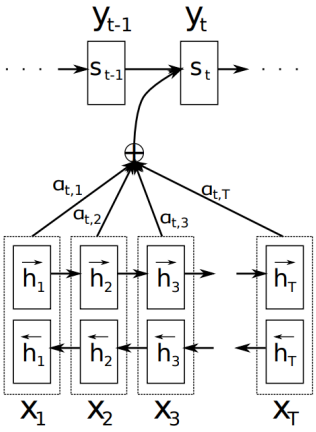


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

2. Effective Approaches to Attention-based Neural Machine Translation

文章的贡献是在 attention 机制的基础上提出了一种改进的模型，更详细地分析了不同的 attention 机制带来的效果。

Attention-based Models

基于注意力机制的模型分为两大类：*global* 和 *local*，主要区别在于“注意力”是在所有位置还是一部分位置上。两者的共同点是，在 decoding 阶段的时刻 t，都将 LSTM 的 top layer 的隐藏状态 $\mathbf{h}_t$ 作为输入，然后根据上下文向量 $\mathbf{c}_t$ 来预测当前目标词 y_t。尽管 $\mathbf{c}_t$ 的推导方法不同，后续的步骤是相同的。

给定隐藏状态 $\mathbf{h}_t$ 和 source-side 上下文向量 $\mathbf{c}_t$，将这些向量组合起来得到 attentional hidden state：

$$\tilde{\mathbf{h}_t} = \tanh(\mathbf{W_c}[\mathbf{c}_t; \mathbf{h}_t])$$

$\tilde{\mathbf{h}_t}$ 输入至 softmax 层，得到预测概率：

$$p(y_t|y_{<t}, x) = softmax(\mathbf{W_s}\tilde{\mathbf{h}_t})$$

下面详细介绍每种模型中 $\mathbf{c}_t$ 是如何计算的。

Global Attention

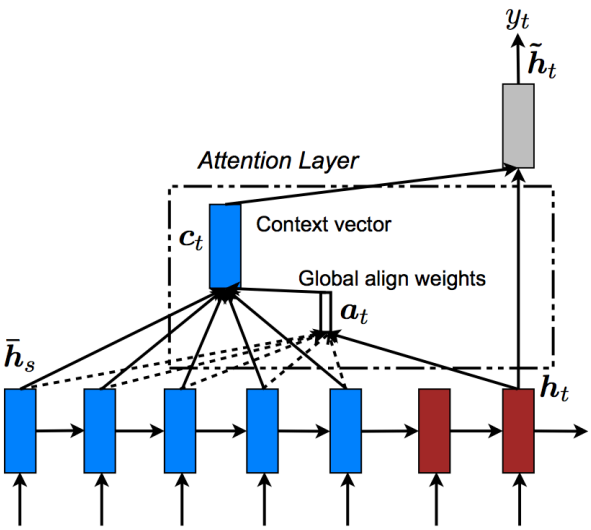
global attentional model 的思想是推导 c_t 时考虑 encoder 的所有隐藏状态。在此模型中，变长 alignment vector $\mathbf{a}_t$ 通过比较当前 target hidden state $\mathbf{h}_t$ 和每个 source hidden state $\bar{\mathbf{h}}_s$ 来得到：

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

这里，score 为 *content-based* 函数，有三种计算方法：

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W_a} \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W_a}[\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s]\right) & \textit{concat} \end{cases}$$

以 alignment vector a_t 作为权重，上下文向量 c_t 为 source hidden states 的加权平均。文章指出，与上篇文章的不同之处在于：一是该文只使用隐藏状态的 top layers，encoder 和 decoder 都是如此；而上篇文章，在双向 encoder 中，使用了前向和后向 source hidden states 的拼接。二是计算流程的不同，本文的计算流程是 $\mathbf{h}_t \rightarrow \mathbf{a}_t \rightarrow \mathbf{c}_t \rightarrow \tilde{\mathbf{h}_t}$，而上篇文章是 $\mathbf{h}_{t-1} \rightarrow \mathbf{a}_t \rightarrow \mathbf{c}_t \rightarrow \mathbf{h}_t$；三是，上篇文章只使用了一种 alignment 函数，*concat* 乘积，而本文提出了更好的方法。



Local Attention

Global attention 有个缺点，即对每个目标词，都要关注所有的 source side 词，这既费时又不太现实，特别是对于长序列，比如段落或文章。为解决这个问题，本文提出了 local attention，对每个目标词，仅关注 source positions 的一个子集。

该模型是 *soft* 和 *hard* attention 的折中，两者用在图像处理中。其中，soft attention 就像 global attention，weights are placed "softly" over all patches in the source image。hard attention 则一个时刻选择图像的一个 patch。

local attention 关注上下文的一个窗口，相比 soft attention，不需要复杂的计算；相比 hard attention，又比较容易训练。具体来说，在时刻 t，对每个目标词生产一个对齐位置 p_t，上下文向量 c_t 通过对窗口 $[p_t - D, p_t + D]$ 内的隐藏状态求加权平均得到，D 为经验值。不同于 global 方法，local 对齐向量 a_t 为固定维度，$\in \mathbb{R}^{2D+1}$。模型的两个变种为：

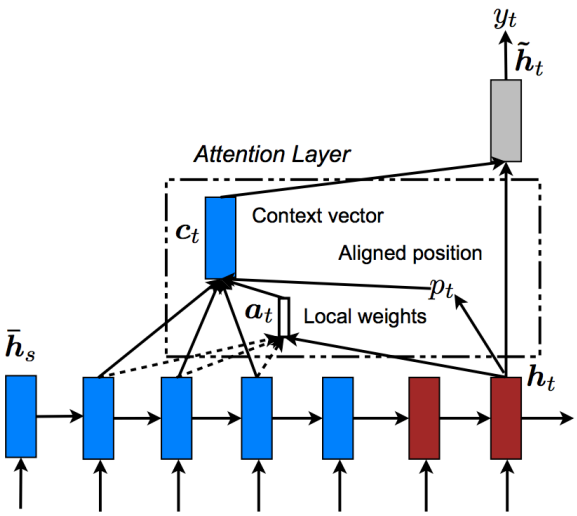
- *Monotonic* alignment — 令 $p_t = t$，即假设源和目标序列是大致单调对齐的，对齐向量 a_t 和上面一样。
- *Predictive* alignment — 预测对齐位置为：

$$p_t = S \cdot sigmoid(\mathbf{v}_p^T \tanh(\mathbf{W_p h}_t))$$

$\mathbf{W_p}$ 和 $\mathbf{v}_p$ 是模型参数，S 为源句子长度，作为 sigmoid 结果，$p_t \in [0, S]$。为了倾向于 p_t 附近的点对齐，使 p_t 周围满足高斯分布：

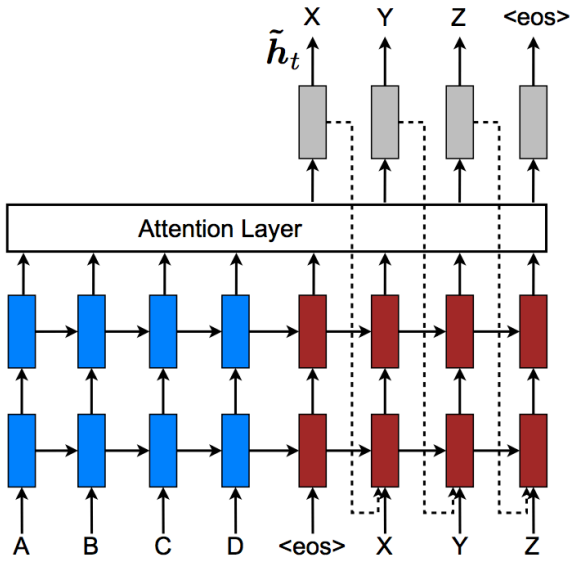
$$\boldsymbol{a}_t(s) = \text{align}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) \exp\left(-\frac{(s - p_t)^2}{2\sigma^2}\right) \quad (10)$$

对齐函数和上面一样，标准差设经验值 $\sigma = \frac{D}{2}$，p_t 为实数，s 为以 p_t 为中心的窗口之间的整数。



Input-feeding Approach

文章提出了 *input-feeding* 方法，attentional vector $\tilde{h_t}$ 与下一时刻的输入做拼接，如图所示：



文章说，这样做的效果有两点：

- we hope to make the model fully aware of previous alignment choices;

- we create a very deep network spanning both horizontally and vertically.

实验结果

模型使用 WMT'14 训练数据，包括 4.5M sentences pairs（116M 英文词汇，110M 德语词汇。）对每种语言，仅用使用频率为前 50K 的词汇，对其它的词汇，使用 <unk> 代替。在训练过程中，过滤掉长度超过 50 个词汇的句子，并且 shuffle mini-batches。stacking LSTM 模型有 4 层，每层有 100 个单元，1000 维的 embedding。训练中的一些细节问题：

- 参数均匀分布为 [-0.1,0.1];
- 使用 plain SGD 训练 10 epochs;
- 学习率开始为 1，5 轮之后，每个 epoch 学习率减半；
- mini-batch 大小为 128；
- 范数超过 5 时，对归一化梯度进行 rescale;
- 以 0.2 的概率进行 dropout。