Background

Elman RNN:

$$h_t = RNN(x_t, h_{t-1}) = \sigma(Wx_t + Uh_{t-1} + b)$$

RNN Decoder:

$$h_t^{dec} = RNN([y_{t-1}; C], h_{t-1}^{dec})$$
$$y_t = softmax(W \cdot h_t^{dec})$$

Bahdanau Attention (Additive Attention):

$$h_t, s_t分别是Encoder, Decoder的时间步隐藏状态$$
$$s_t = f(y_{t-1}, s_{t-1}, c_t)$$
$$c_t = \sum_{j=1}^{n} a_{tj} h_j$$
$$a_{tj} = softmax(\epsilon_{tj}) = \frac{exp(\epsilon_{tj})}{\sum_{k=1}^{n} exp(\epsilon_{tk})}$$
$$\epsilon_{tj} = a(s_{t-1}, h_j)$$
$$a(s_{t-1}, h_j) = V^T \cdot tanh(W_a \cdot s_{t-1} + U_a \cdot h_j)$$

additive attention 是后人总结的概念，指注意力分数通过Query和Key通过加法混合，且通常搭配tanh激活函数，即

$$a(s_{t-1}, h_j) = V^T \cdot tanh(W_a \cdot s_{t-1} + U_a \cdot h_j)$$

当然Query和Key也是后人总结的，ϵ_{tj}其实也不是最终的注意力分数（还要经过softmax归一化），事实上Bahdanau也没用注意力分数这个词。

Bahdanau等人把$a(\cdot)$叫做$alignment\ model$,对齐模型，该模型通常是一个浅层神经网络（如单层MLP），学习对齐规则。对齐的核心思想可以表述为：在结构不同但内容相关的信息之间建立映射，通常是学习语义之间的相关性的强弱。

Framework

Embeddings

> Similarly to other sequence transduction models, we use learned embeddings to convert the input tokens and output tokens to vectors of dimension d_{model}. We also use the usual learned linear transformation and softmax function to convert the decoder output to predicted next-token probabilities. In our model, we share the same weight matrix between the two embedding layers and the pre-softmax linear transformation, similar to [30]. In the embedding layers, we multiply those weights by $\sqrt{d_{model}}$

$$e = f(idx) = E[idx]$$
$$output = \frac{e}{\sqrt{d_{model}}}$$

$E[idx]$一般表示矩阵E的第idx行，虽然$f(idx)$是个离散函数，不可导，

但$\nabla L(f(idx))$损失函数对矩阵中的权重是可导的，所以该权重是可以 通过学习得到的（learned）

Position Encoding

$$PE(pos, 2i) = sin(pos/10000^{2i/d_{model}})$$
$$PE(pos, 2i+1) = sin(pos/10000^{2i/d_{model}})$$
$$output = x + PE$$

d_{model}指的是嵌入层的维度，例如之前把"hello"映射成了一个512维的向量，d_{model}就是512

注意公式中$2i, 2i+1$隐含了步长为2的信息, output的维度也是d_{model}

Self Attention, Multi-Head, Mask and Cross

$$Q = XW_q$$
$$K = XW_k$$
$$V = XW_v$$
$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$
$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_n)W^O$$
$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$
$$padding\ mask: mask(x) = \begin{cases} -inf, is\ padding \\ 0, else \end{cases}$$
$$look\ ahead\ mask: mask(i) = \begin{cases} -inf, i > current_pos \\ 0, else \end{cases}$$

$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$就是Scaled Dot Attention, 除以$\sqrt{d_k}$就是scaled

多头注意力首先用多个头分别理解整个序列，然后产生了一个特征序列，各特征信息在该序列中分块存在，然后通过应用W^O分发到整个序列。

Cross Attention指Decoder（的每个模块）中有一种多头注意力器是K和V是由Encoder的X经变换得到的。事实上，Encoder是由N个编码器模块串联而成，由最后一个模块的末端输出X分发给Decoder中所有（一般同样是N个）解码器模块。

KQ点积，也许可以这样子理解，语义相关性强的K和Q在高维空间中应该是方向相近的，如果模型能把模长约束在一定范围内，甚至可以说是位置相近的，这时候点积在数值上就会比较大。

Feed Forward, Residual Connections and Add&Norm

> In addition to attention sub-layers, each of the layers in our encoder and decoder contains a fully connected feed-forward network, which is applied to each position separately and identically. This consists of two linear transformations with a ReLU activation in between.

原文相当明白。

Shifted Right and Teacher Forcing

首先，在经典的Transformer中，训练时，会把目标序列右移一位后把第一位设为$< Start >$或者$< sos >$之类的占位符，这叫Shifted Right，这是自回归推理的基础。同时，训练时，这个Shifted Right目标输出序列会被强制设定为Decoder的输入，这叫做教师强制，然后，Decoder能同时对各个位置进行学习（也存在前瞻掩码）。

等到推理时，类似于RNNdec的逐步生成，Decoder的输出会成为自己下一步的输入，这就是自回归推理，上一步和下一步是串行的，但是同一步内，Decoder对之前所有时间步的分析是并行的，同时，这个过程存在look ahead mask，前瞻掩码，又叫因果掩码，令Decoder只能分析这一步之前的时间步，而不能分析未来的时间步。