Week4

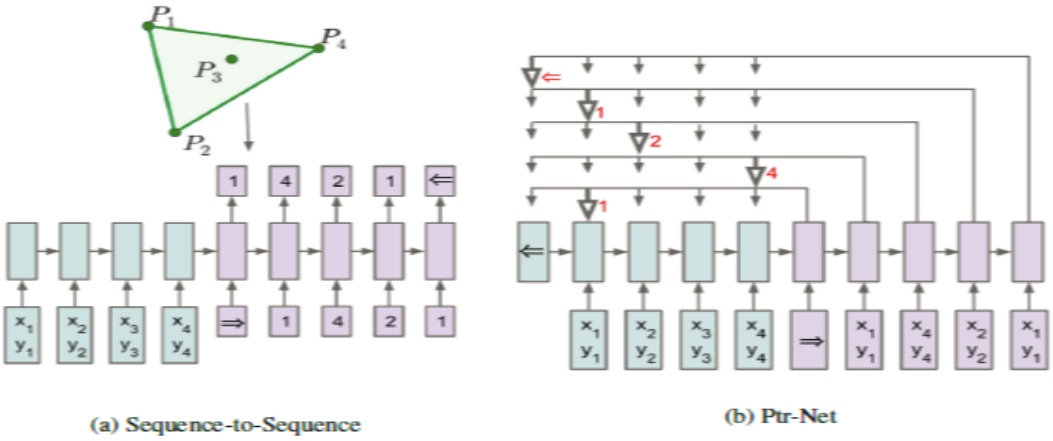
1.Pointer Networks 简介

凸包(convex hull)：在一个多边形边缘或内部任意两个点的连线都包含在多边形边界或者内部中，即包含点集合 S 中所有点的最小凸多边形称为凸包。

传统的 seq2seq 模型无法解决输出序列的词汇表会随着输入序列长度的改变而改变的问题的，如寻找凸包。

如下图 a 的例子是给定 p1 到 p4 四个二维点的坐标，要求找到一个凸包，显然答案是 p1->p4->p2->p1。图 a 是传统 seq2seq 模型的做法，就是把四个点的坐标作为输入序列输入进去，然后提供一个词汇表：[start, 1, 2, 3, 4, end]，最后依据词汇表预测出序列[start, 1, 4, 2, 1, end]，对于图 a 的传统 seq2seq 模型来说，它的输出词汇表已经限定，当输入序列的长度变化的时候（如变为 10 个点）它根本无法预测大于 4 的数字。

对于寻找凸包这类问题，输出往往是输入集合的子集。基于这种特点，论文作者考虑能不能找到一种结构类似编程语言中的指针，每个指针对应输入序列的一个元素，从而我们可以直接操作输入序列而不需要特意设定输出词汇表，给出的答案是指针网络（Pointer Networks）。图 b 中的 Pointer Networks，它预测的时候每一步都找当前输入序列中权重最大的那个元素，而由于输出序列完全来自输入序列，它可以适应输入序列的长度变化。



根据传统的注意力机制（见下图），得到的 attention 评分正是针对输入序列的权重，完全可以把它拿出来作为指向输入序列的指针，在每次预测一个元素的时候找到输入序列中权重最大的那个元素。

$$
\begin{aligned}
u_j^i &= v^T \tanh(W_1 e_j + W_2 d_i) \quad j \in (1, \ldots, n) \\
a_j^i &= \mathrm{softmax}(u_j^i) \quad j \in (1, \ldots, n) \\
d_i' &= \sum_{j=1}^{n} a_j^i e_j
\end{aligned}
$$

其中ej是encoder的隐状态，而di是decoder的隐状态，v,W1,W2都是可学习的参数，在得到 uj之后对其执行softmax操作即得到 aj。

于是作者就按照这个思路对传统注意力机制进行了修改和简化，公式变成了这个样子：

$$
\begin{aligned}
u_j^i &= v^T \tanh(W_1 e_j + W_2 d_i) \quad j \in (1, \ldots, n) \\
p(C_i | C_1, \ldots, C_{i-1}, \mathcal{P}) &= \mathrm{softmax}(u^i)
\end{aligned}
$$

Pointer Networks 直接将 softmax 之后得到的 attention 得分当成了输出，让其承担指向输入序列特定元素的指针角色。

总结：传统的带有注意力机制的 seq2seq 模型是，先使用 encoder 部分对输入序列进行编码，然后对编码后的向量做 attention，最后使用 decoder 部分对 attention 后的向量进行解码从而得到预测结果。但是作为 Pointer Networks，得到预测结果的方式便是输出一个概率分布 attention 评分，也即所谓的指针。换句话说，传统带有注意力机制的 seq2seq 模型输出的是针对输出词汇表的一个概率分布，而 Pointer Networks 输出的则是针对输入文本序列的概率分布。

其实我们可以发现，因为输出元素来自输入元素的特点，Pointer Networks 特别适合用来直接复制输入序列中的某些元素给输出序列。

以上内容基本来自参考文献里的内容，写入笔记中，只为加深个人对 Pointer Networks 的学习理解。关于 Pointer Networks 应用的内容也可详见参考文献里的文献。此文献对课件里的 Pointer Networks 内容均有较为详细的阐述，为 Pointer Networks 非常好的学习资料。

参考文献：

1.Pointer Networks 简介及其应用， Tworld(coding)， 2020，

https://zhuanlan.zhihu.com/p/48959800