Pointer Networks

1. 传统的Seq2Seq模型

Seq2Seq模型是RNN的一个重要的应用场景，它实现了把一个序列转换成另一个序列的功能，并且不需要输入序列和输出序列等长。

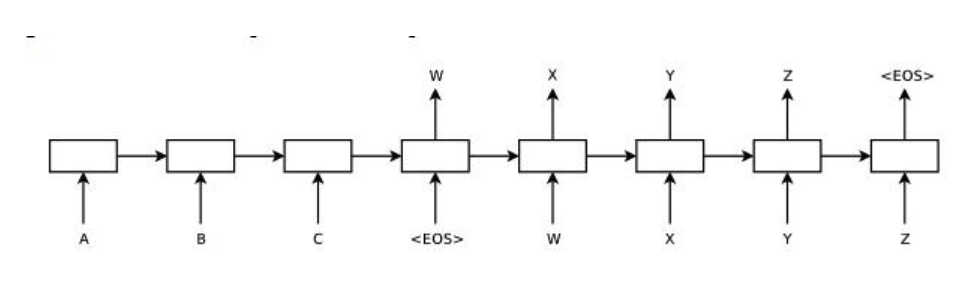


图1： Seq2Seq模型结构

在这幅图中，模型把序列“ABC”转换成了序列“WXYZ”。分析其结构，我们可以把seq2seq模型分为encoder和decoder两部分。Encoder部分接受“ABC”作为输入，然后将这个序列转换成一个中间向量C，向量C可以认为是对输入序列的一种理解和表现形式。然后decoder部分把中间向量C作为自己的输入，通过解码操作得到输出序列“WXYZ”。

后来加入的Attention Mechanism使得seq2seq模型的性能大幅提升。Attention Mechanism的作用就是讲encoder的隐状态按照一定的权重加和之后拼接（或者直接加和）到decoder的隐状态上，以此作为额外的信息，起到所谓的软对齐作用。

1. Pointer Networks

Pointer Networks是对Attention Mechanism简化得到的。作者开篇就提到了，传统的seq2seq模型是无法解决输出序列的词汇随着输入序列长度的改变而改变的问题(因为传统的seq2seq模型的输出基本是词汇表大小的一个向量，表示词的分布概率)。

基于这种特点，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，作者给出的答案就是指针网络（Pointer Networks）。

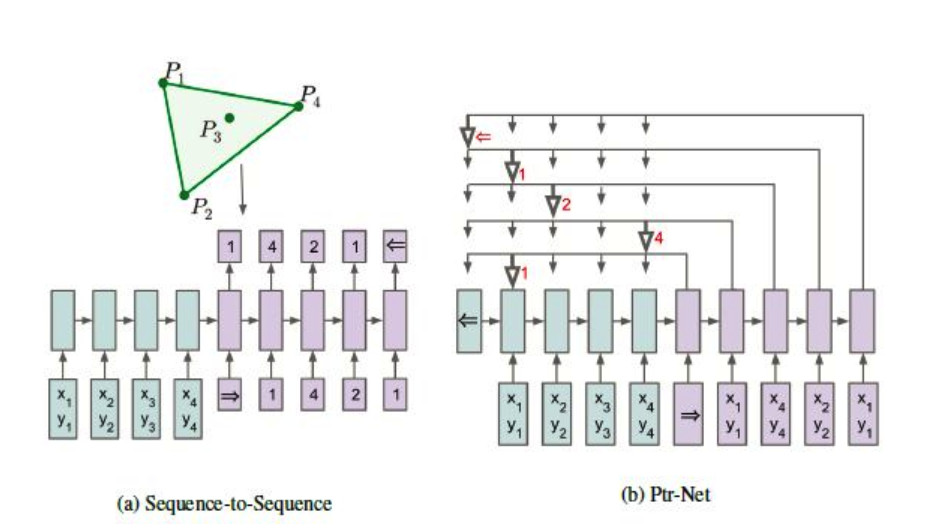


图2：Pointer Networks实例

这个图的例子是给定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是传统的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输出序列为[start, 1,4,2,1,end]，缺点作者也提到了，对于图a的传统seq2seq模型来说，它的输出词汇表已经限定了，当输入序列的长度变化时（如变成10个点），他就没法预测大于4的数字。

图b是作者提出的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。

那么Pointer Networks是如何实现的呢？

我们首先看下传统的注意力机制的公式：

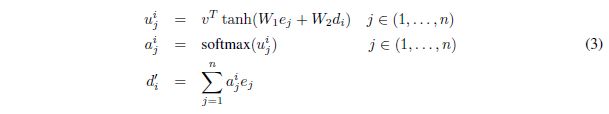


图3：传统的注意力机制公式

其中e是encoder的隐状态，d是decoder的隐状态，v,w1,w2都是可学习的参数，在得到之后对其进行执行softmax操作即可得到。这里的就是分配给输入序列的权重，依据该权重求加权和，然后把得到的拼接到（或者加和）到decoder的隐状态上。最后让decoder部分根据拼接后新的隐状态进行解码和预测。

根据传统的注意力机制，作者想到，所谓的针对输入序列的权重，完全可以把它拿来当做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就可以了。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，公式就变成了如下：

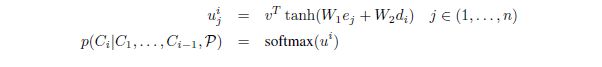


图4：Pointer Network的公式

第一个公式和之前没有区别，然后第二个公式则是说Pointer Networks直接将softmax之后得到的当成了输出，把他当做指向输入序列特定元素的指针角色。

所以总结一下，传统的带有注意力机制的seq2seq模型的运行过程是先使用encoder部分对输入序列编码，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，也就是所谓的指针。**换句话说，传统的带有注意力机制的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一个概率分布。**