**第十章读书报告**

09118223 吴亦珂

读书进度：第十章读完

1. **问题列表**

（我提出）P6页“For neural generation, where we are trying to generate novel outputs, we can simply sample from the softmax distribution. However, for applications like MT where we’re looking for a specifific output sequence, random sampling isn’t appropriate and would likely lead to some strange output。”这里为什么MT会有一些不好的结果。

讨论结果：neural generation的目标只是要产生一个句子，而机器翻译要考虑到上下文以及因此就会有更多限制，需要一个完整的序列。因此这种方法对于机器翻译可能并不是一个好方法。

（我提出）Beam search上一段中最后一句话说这个方法"invariant doesn't hold"具体指什么意思吗？

讨论结果：应该值由于动态规划算法的原因，所以对维特比算法造成了一定的影响。

（我提出）attention机制是否就是指通过给予编码器和解码器一定权重从而使得机器可以更有针对性的学习？  
讨论结果：attetion可以通过向量积的形式找到对于接下来一步翻译最重要的原文部分，这样就为每一步翻译有效的提供了其需要的上下文，而且信息缺失也很少。

（别人提出）Attentio机制与直接上下文之间的比较，Attention是否存在缺点。  
讨论结果：因为是对序列的所有元素并行处理的，所以无法考虑输入序列的元素顺序，这在自然语言处理任务中比较糟糕。因为在自然语言中，语言的顺序是包含了十分多的信息的，如果缺失了该部分的信息，则得到的结果往往会大大折扣。

（别人提出）Self-Attention and Transformer Networks这两个概念是指什么？是否是Attention的延申。  
讨论结果：应该可以看做attention的延伸。 self-attention显然是attentio机制的一种。上面所讲的attention是输入对输出的权重，例如在上文中，是I am a student 对学生的权重。self-attention则是自己对自己的权重，例如I am a student分别对am的权重、对student的权重。之所以这样做，是为了充分考虑句子之间不同词语之间的语义及语法联系。

Transformer可以视为一个升级版的seq2seq，由一个encoder和一个decoder组成，但是encoder和decoder都不用RNN，而是换成了多个attention。

1. **读书收获**

**10.2 Encoder-Decoder Networks**

计算输出层的方法：可以仅仅依赖于刚刚计算出来的隐藏状态。另一种方法是计算取决于新计算出的隐藏状态，之前状态的输出以及encoder context。

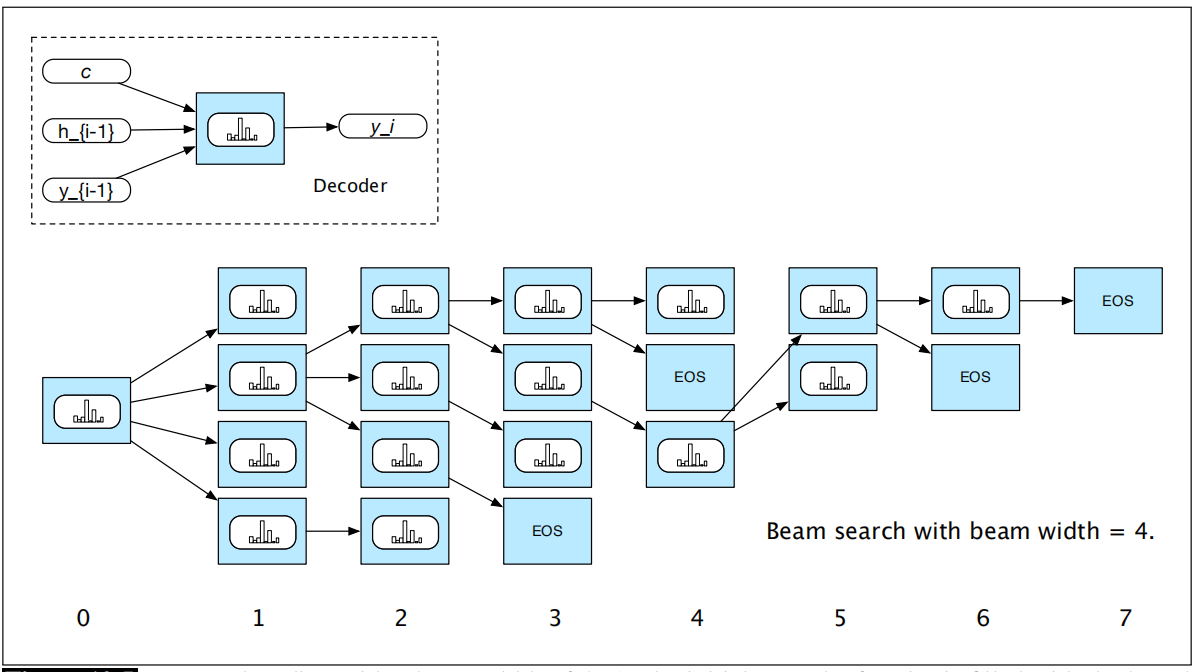


Beam search：解决解码问题可以使用beam search方法。

首先，根据每个选项的softmax概率给予相应得分，并且根据得分选择B个最好的选项。这些选项构成search frontier。这些搜索路径对应的序列称为hypotheses。

在随后的步骤中，每一个hypotheses都会增长通过传递给不同的解码器并且会再次生成整个词表的softmax。每一个hypotheses必须包括最新产生的单词，context vector，以及之前步骤产生的隐藏状态。每一个新的假设用概率的乘积P（yi | y<i）评分，即当前单词选择的概率乘以通向它的路径的概率。为了控制边界的指数增长，它被修剪成只包含做好的B个假设。

这个过程一直持续到生成一个<\s>来表示已经找到一个完整的候选项输出。



这样产生的一个较为复杂的问题是不同的假设可能长度不同，由于计算方案的原因，较长的假设比较短的假设得分要低。解决方法是进行标准化。

之前定义了上下文向量c作为编码器隐藏状态的函数。不幸的是，隐藏状态的数量随输入改变，很难直接将它们用作解码的上下文。前面描述的基本方法避免了这个问题，因为c只是编码的最后隐藏状态。这个方法可以简单而且将context变为了固定的长度。但是，不可避免使得注意集中在语句的最后而不是全部。

解决方法可以使用Bi-RNNs，或者另一种方法是简单地求和或平均编码器生成上下文向量的隐藏状态，但这种方法会损失一些有用的信息。

**10.3 Attention**

为了克服注意集中在句子尾端的机制，可以使用注意力的机制。

第一步是根据编码层的隐藏状态动态生成向量替代原来的静态向量。

计算ci的第一步是计算一个得分向量去表示每个编码器隐藏状态与解码器状态的相关性。

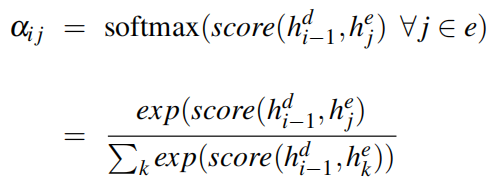
可以使用两个向量的点积来表示：



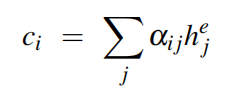
虽然简单的点积可以是有效的，但它是一个静态的度量，不能在训练过程中适应应用的特点。一个更加鲁棒的方法是增加一个权重。

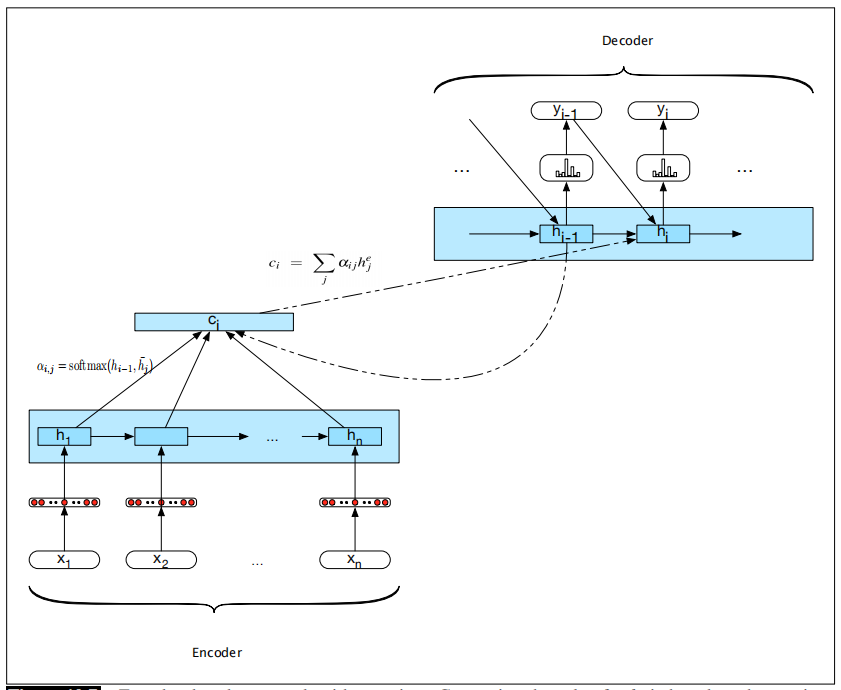


通过基于权重，可以给予网络根据应用特点学习哪个重要的能力。为了使用这些分数，可以使用softmax操作。



最后，我们可以计算出给定长度的context vector：





**10.4 Applications of Encoder-Decoder Networks**

可以用于摘要，句子简化，问答系统，以及图片裁剪。