信息网络专题研究课文献阅读报告（应用层）

一、 文献信息

1、 作者：• Ashish Vaswani • Noam Shazeer • Niki Parmar • Jakob Uszkoreit • Llion Jones • Aidan N. Gomez • Lukasz Kaiser • Illia Polosukhin

2、 题目：Attention Is All You Need

3、 发表途径：Google

4、 发表时间：NeurIPS 2017

二、 问题意义

1. 研究背景：

机器翻译是利用计算机将一种自然语言(源语言)转换为另一种自然语言(目标语言)的过程。它是 NLP 的一个分支，在促进政治、经济、文化交流等方面起到越来越重要的作用。深度学习做 NLP 的方法，一般先将句子分词，然后每个词转化为对应的词向量序列。每个句子都对应的是一个矩阵，这样问题就变成了编码这些序列了。

主流的机器翻译序列建模都是基于的循环(RNN)或卷积(CNN)神经网络。RNN 递归式进行，结构简单，适合序列建模，但无法并行，速度较慢。为了应对 RNN 无法并行问题，发展了 CNN 的解决方案，方便并行，但它只能捕捉到局部信息，可以通过层叠来增大感受野，较为复杂。

2. 研究问题：

RNN 要逐步递归才能获得全局信息，速度慢无法实现并行；CNN 只能捕捉到局部信息，但是可以通过层叠来增大感受野。这篇文章要创建一种新的思路，能够获取了全局信息，实现并行运算，提高计算速度。

3. 研究意义：

针对 NLP 里的机器翻译问题，提出了一种被称为“Transformer”的网络结构，Transformer 是第一个完全依靠 self-attention 的转换模型来计算其输入和输出的表示，而不使用序列对齐的 RNN 或卷积的模型。自注意力机制（Self-attention）能够把输入序列上不同位置的信息联系起来，然后计算出整条序列的某种表达，目前自注意力机制主要应用于阅读理解、提取摘要、文本推论等领域。这样做最大好处是能够并行计算，提高计算速度。论文的主要贡献之一是它表明了内部注意力在机器翻译的序列编码上是相当重要的，而之前关于 Seq2Seq 的研究基本都只是把注意力机制用在解码端。

三、 思路方法

大多自然语言转换模型都包含 encoder-decoder 结构，encoder 负责将输入的离散符号序列映射成连续值序列。decoder 负责生成一个输出符号序列。之前生成的输出会作为额外的输入，用于生成下一个输出。Transformer 遵循这种整体架构，在编码器和解码器部分使用栈的 self-attention 和各数据独立的全连接层。

Transformer 由且仅由 self-Attenion 和 Feed Forward Neural Network 组成。编码某个单词时，self-attention 模块帮助结合其他单词的信息；feed-forward network 在不同单词之间共享。一个基于 Transformer 的可训练的神经网络可以通过堆叠 Transformer 的形式进行搭建，作者的实验是通过搭建编码器和解码器各 6 层，总共 12 层的 Encoder-Decoder。

1. 编码器和解码器栈

Transformer 的本质上是一个 Encoder-Decoder 的结构。 编码器由 6 个编码 block 组成，同样解码器是 6 个解码 block 组成。与所有的生成模型相同的是，编码器的输出会作为解码器的输入

a) 编码器 Encoder

Transformer 模型的 Encoder 由 6 个基本层堆叠起来，每个基本层包含两个子层，第一个子层是一个注意力机制，第二个是一个全连接前向神经网络。对两个子层都引入了残差边以及 layer normalization。

b) 解码器 Decoder

Transformer 模型的 Decoder 也由 6 个基本层堆叠起来，每个基本层除了 Encoder 里面的那两个以外，Decoder 它和 encoder 的不同之处在于 Decoder 多了一个 Encoder-Decoder Attention，两个 Attention 分别用于计算输入和输出的权值，同样引入残差边以及 layer normalization。

2. 注意力机制

Decoder 比 encoder 的不多了一个 Attention。注意力机制（Attention）简单来说就是给定一个查找（query）和一个键值表（key-value pairs)，将 query 映射到正确的输入的过程。此处的 query、key、value 和最终的输出都是向量。输出往往是一个加权求和的形式，而权重则由 query、key 和 value 决定。

a) 放缩点积 attention（scaled dot-Product attention）

常用的使用点积进行相似度计算的 attention，只是多除了一个（为 K 的维度）起到调节作用，使得内积不至于太大。Attention 的计算方法，整个过程可以分成 7 步：

1) 将输入单词转化成嵌入向量；
2) 根据嵌入向量得到 q ，k ，v 三个向量；
3) 为每个向量计算一个 score： score = q×k ；
4) 为了梯度的稳定，Transformer 使用了 score 归一化，即除以 $\sqrt{d_k}$ ；
5) 对 score 施以 softmax 激活函数；
6) softmax 点乘 Value 值 v ，得到加权的每个输入向量的评分 v ；
7) 相加之后得到最终的输出结果 z ：$z = \sum v$。

b) 多头 attention（Multi-head attention ）

这个是 Google 提出的新概念，是 Attention 机制的完善。将 self-attention 做多次后，进行拼接，点乘一个训练得到的矩阵 WO,得到 self-attention 输出。

Query，Key，Value 先经过一个线性变换，输入到放缩点积 attention，做 h 次，也就是所谓的多头。每次 Q,K,V 进行线性变换的参数 W 不一样。然后将 h 次的放缩点积 attention 结果进行拼接，再进行一次线性变换得到的值作为多头 attention 的结果。多头 attention 的不同之处在于进行了 h 次计算而不仅仅算一次，好处是可以允许模型在不同的表示子空间里学习到相关的信息。

Multi-Head Attention 的输出分成 3 步：

1) 将数据 X 分别输入到多个 self-attention 中，得到多个加权后的特征矩阵。
2) 将多个特征矩阵按列拼成一个大的特征矩阵；
3) 特征矩阵经过一层全连接后得到输出。

c) 模型中 Attention 的应用

Transformer 中用了三种不同的方式使用 multi-head attention：

1) Encoder 模块的 Self-Attention，每层的 Self-Attention 的输入是上一层的输出。Encoder 中的每个 position 都能够获取到前一层的所有位置的输出

2) Decoder 模块的 Mask Self-Attention，在 Decoder 中，每个 position 只能获取到之前 position 的信息，因此需要做 mask。

3) Encoder-Decoder 之间的 Attention，其中 Q 来自于之前的 Decoder 层输出， K,V 来自于 encoder 的输出，decoder 的每个位置都能获取输入序列的所有位置信息。

3. 位置前馈网络（Position-wise Feed-Forward Networks）

除了 attention 子层之外，编码器和解码器中的每个层都包含一个完全连接的前馈网络，对每个 position 的向量分别进行相同的操作。 该前馈网络包括两个线性变换和一个 ReLU 激活输出。

4. 词嵌入和 softmax（Embeddings and Softmax）

与其他序列转换模型类似，采用词嵌入将输入符号和输出符号转换为向量。解码的特征向量经过一层激活函数为 softmax 的全连接层之后得到反映每个单词概率的输出向量，预测的下一个字符的概率。

5. 位置编码(Positional Encodingl)

Transformer 模型不包含循环网络和卷积网络，为了使模型能够利用序列的位置信息，具有捕捉顺序序列的能力，必须加入关于序列中字符的相对或绝对位置的一些信息。论文中在编码词向量时引入了位置编码（Position Embedding）的特征。具体地说，位置编码会在词向量中加入了单词的位置信息，使 Transformer 能区分不同位置的单词。

四、 实验结果

使用自注意力机制的原因具有很多优点，论文中提到主要从三个方面考虑

1) 每一层的复杂度：采用自注意力机制复杂度低。

论文中给出了和 RNN，CNN 计算复杂度的比较。如果输入序列 n 小于表示维度 d 的话，每一层的时间复杂度 self-attention 是比较有优势的。当 n 比较大时，作者也给出了一种解决方案 self-attention（restricted）即每个词不是和所有词计算 attention，而是只与限制的 r 个词去计算 attention。

2) 是否可以并行：该模型可以并行化的计算

在并行方面，多头 attention 和 CNN 一样不依赖于前一时刻的计算，可以很好的并行，优于 RNN。

3) 网络中远程依赖之间的路径长度：长距离依赖学习。

影响长距离依赖关系的能力的关键因素是前向和后向信号在网络中传播路径的长度。输入输出序列之间的路径越短，长距离依赖关系就越容易得到。在长距离依赖上，由于 self-attention 是每个词和所有词都要计算 attention，所以不管他们中间有多长距离，最大的路径长度也都只是 1，因此容易形成长距离依赖关系。

五、 启发思考

Transformer 具有很多优点文中也有所提及：设计创新，抛弃了 RNN 或者 CNN，效果不错； Transformer 将任意两个单词的距离是 1，有效解决 NLP 中长期依赖问题；是非常有科研潜力的一个方向可以应用在很多地方；算法的并行性非常好。但同时我在网上学习后发现 Transformer 也存在一定缺陷：完全抛弃 RNN 和 CNN 使模型失去了捕捉局部特征的能力；失去的位置信息其实在 NLP 中非常重要，而论文中在特征向量中加入 Position Embedding 也

只是一个权宜之计，并没有改变 Transformer 结构上的固有缺陷。

我还运行了一下论文配套的代码，我本来想运行了 tensorflow 版本的，但是论文附加的代码比较复杂，我的电脑运行有点困难，所以我在网上找了一个简单明了的版本。

步骤 1. 运行以下命令，下载 IWSLT 2016 德语-英语并行语料库。bash download.sh

步骤 2. 运行以下命令以创建预处理的列车/评估/测试数据。python prepro.py

步骤 3. 运行以下命令。python train.py

步骤 3. 或下载预训练的模型。

下面是运行得到的 Bleu score on devset 、Training Loss、Learning rate 的三个图。