语言分析与机器翻译
课程报告

姓　　名	高成浩	学　　号	2101711
班　　级	计硕 2102	指　导　教　师	朱靖波
实　验　名　称	语言分析与机器翻译课程报告		
开　设　学　期	2021 -2022 第一学期		
开　设　时　间	第 2 周——第 9 周		
报　告　日　期	2021 年 11 月 26 日		
评　定　成　绩		评　定　人	
		评　定　日　期	

计算机科学与工程学院

一、任务要求

基于 Transformer 模型实现，使用预处理好的 IWSLT'14 De-En 数据集进行训练，输入德文输出对应的英文。

二、Transformer 模型结构

为了解决历史信息在 RNN 或是 LSTM 单元传递中衰减的问题，谷歌的研究人员提出了一种全新的模型 Transformer [错误!未找到引用源。]。该模型不采用任何循环单元或是卷积操作，而是采用注意力机制直接捕获长距离序列中任意两个单元之间的关系，并且该方法非常适合在 GPU 上进行并行化操作，能大大提升模型训练速度。下面介绍 Transformer 模型结构中的核心部分，其详细结构如图 2.1 所示，编码器和解码器可根据需要进行不同层数堆叠。

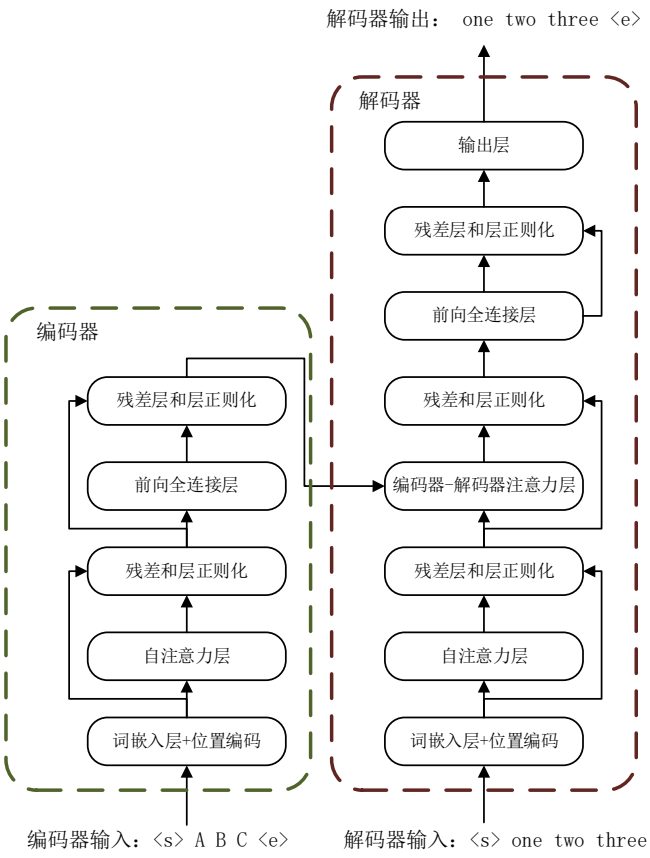


图 2.1 Transformer 模型结构

1）位置编码及词嵌入层

由于语言具有顺序的特点，模型在处理序列输入时需要考虑每个词的顺序关系，Transformer 引入位置编码对每个词添加额外的位置信息。位置编码方式有多种，在 Transformer 模型中使用不同频率的正余弦函数，如公式（2-1）和（2-2）所示。其中，$PE(\cdot)$表示位置编码函数，pos表示单词位置，i表示位置编码向量中的第几维，d_{model}为模型的一个基础参数，为隐藏层大小，与词嵌入的维度相同。将位置编码与词嵌入相加作为模型的输入。

词嵌入（Word Embedding），是一种单词的分布式表示方式，每个单词不再是正交的 0-1 向量，而是多维实数空间中的一点，也可以看作是欧式空间中的一点。这种表示方式的

$$PE\left(pos,2i\right)=sin\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{2-1}$$

$$PE\left(pos,2i+1\right)=cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{2-2}$$

优点是单词可进行计算和学习，可通过不同计算方式体现不同单词之间的相关性。其编码形式如图 2.2 所示，从图中可以看出使用词嵌入可以利用多个属性表示每个字的信息。

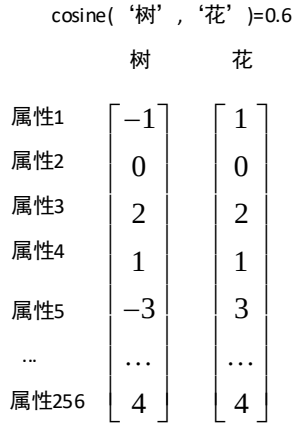


图 2.2 词的分布式表示（词嵌入）

2）注意力层

Transformer 通过一种基于点乘的注意力机制直接捕获序列中任意两个单元

之间的关系，该方法也称为点乘注意力。在计算过程中包含三个重要的参数，分别是 Query，Key 和 Value，简记为 Q，K，V。在编码器及解码器的自注意力层中这三者均为编码器输入序列的向量矩阵，在解码器-编码器注意力层中 Q 为解码器输入，K 和 V 为编码器输出。其具体

的计算过程如公式（2-3）所示，其中 Mask 用于对序列进行补齐操作和对未来词进行屏蔽操作，Softmax 函数用于对注意力矩阵的每一行进行归一化，形成注意力权重矩阵。

$$Attention\left(Q,K,V\right)=Softmax\left(\frac{QK^{T}}{\sqrt{d_{k}}}+Mask\right)V \qquad (2\text{-}3)$$

通过注意力运算，整个序列可表示为 2-D 的矩阵，比 LSTM 网络的固定向量能够传递更多的信息[2]。同时，为了使模型能够对更高维度的语意信息进行提取，Transformer 模型还是用了多头注意力机制，该机制允许模型在不同位置同时关注来自不同表征子空间的信息。该机制计算过程如下：首先将 Q、K 和 V 各自通过一个矩阵W_i^Q，W_i^K和W_i^V转映射到d_q、d_k和d_v,然后对转换后的三个矩阵执行注意力计算，如公式（2-4）所示，最后将计算得到的多个注意力结果进行拼接，并再次映射回原维度，如公式（2-5）所示。其中 h 为多头注意力机制的头数，$d_k=d_k=d_v=d_{model}/h$。

$$head_i=Attention(QW_i^Q,KW_i^K,VW_i^V) \qquad (2\text{-}4)$$

$$MultiHead(Q,K,V)=Concat(head_1,...,head_h)W^o \qquad (2\text{-}5)$$

3）输出层

每个编码器和解码器子层的末端都为一个全连接的前馈神经网络，其中包含两个线性转换和一个 ReLu 激活函数。其公式如（2-6）所示。

$$\mathrm{FFN(x)=max(0,xW_1+b_2)W_2+b_2} \qquad (2\text{-}6)$$

与其它序列转换模型类似，Transformer 可以使用学习好的嵌入层将输入和输出转换为固定维度的向量，也可以使用可学习的线性转换器和 SoftMax 层作为

模型全部解码器子层之后的输出层，进而获取到下一个字符的概率分布。

三、Transformer 模型的改进

由 Vaswani A[1]等人提出的 Transformer 具备调参苦难、收敛速度较慢等问题。因此，Xiong R[2]等人提出新的改进结构：将 Transformer 中 LayerNorm 层放到 Residual 层之前（称之为 Pre-LN Transformer，如图 3.1（b）所示），并且通过实验证明了这种模型结构在调参上甚至不需要 Warm-up 策略，而且可以减少部分训练时间。

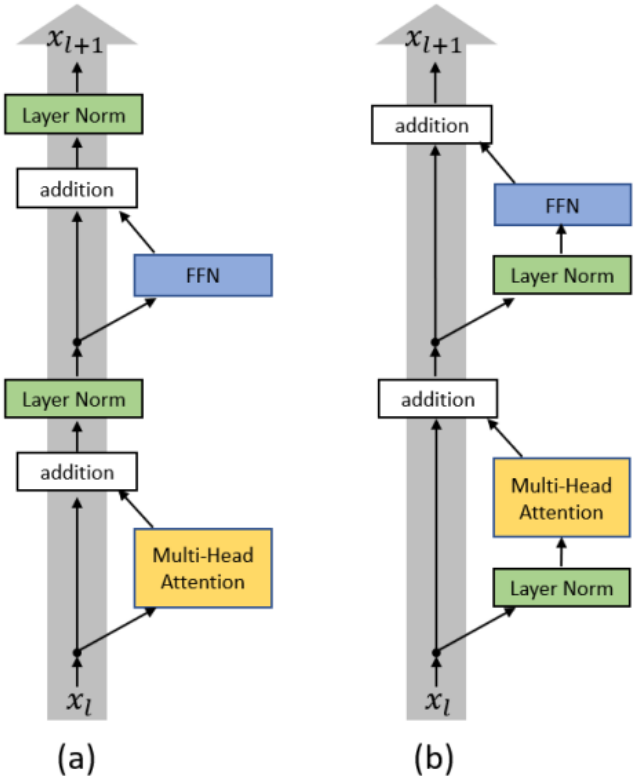


图 3.1 Post-LN Transformer 与 Pre-LN Transformer

四、实验配置

1） 实验环境

本实验使用的实验配置如下：

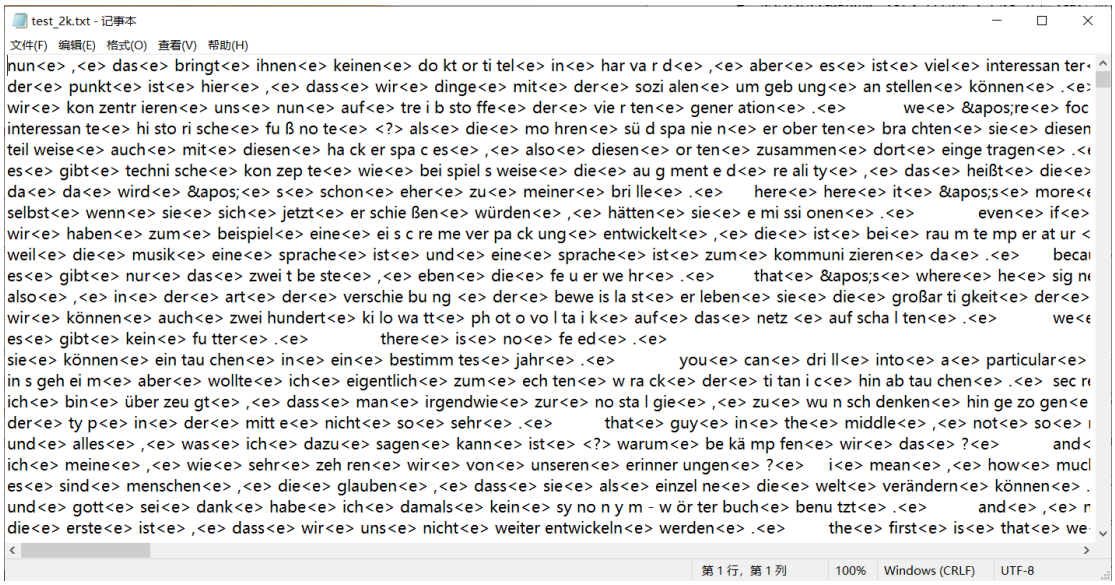
- CentOS 7.3 操作系统、PyTorch 框架

- NVIDIA GeForce GTX TITAN X 12G 开启 GPU 加速
- Intel(R) Core(TM) i7-5930K CPU @ 3.50GHz
- 内存 64G

2） 数据集

本实验使用的数据集为 iwslt'14 de-en，对其进行预处理，下载地址为：https://www.aliyundrive.com/s/jEqtJPU7RgL

后文具体实验方案中需要使用不同的词典大小测试模型，因此使用 BPE 算法生成 2k 和 5k 的词典，再重新生产训练集、验证集和测试集。对于每个单词结尾部分添加"<e>"作为标志，且形式如图 4-1 所示。



3） 评估指标

BLEU（Bilingual Evaluation Understudy）是目前在文本生成领域应用的最广泛的自动评价指标。BLEU 采用 n-gram 匹配的方法评定生成文本与参考原文之间的相似度，即计算生成文本中 n-gram 在参考答案中的匹配率，可用公式（4-1）表示，其中$Count_{hit}$表示生成文本中 n-gram 在{参考原文中命中次数，在生成文本中命中次数的}中间取最小值，$Count_{output}$表示生成文中中共有多少 n-gram。生成文本整体的正确率为各 n-gram 的加权平均，可用公式（4-2）表示。

$$P_n = \frac{Count_{hit}}{Count_{output}} \quad (4\text{-}1)$$

$$P_{avg} = exp\left(\sum_{n=1}^{N} w_n \cdot log\ P_n\right) \quad (4\text{-}2)$$

在训练过程中使用交叉熵对模型的收敛程度进行评估，假设概率分布 p 为期望输出，概率 q 为实际输出，$H(p,q)$ 为交叉熵，通过公式（4-3）进行计算。

$$H(p,q) = -\sum_{x}(p(x)log(q(x)) + (1-p(x))log(1-q(x))) \quad (4\text{-}3)$$

五、实验方案

1）Transformer 改进前后对比实验

依据参考文献[2]所述，使用改进后模型可以简化调参过程，并且可以提高收敛速度。因此，使用相同的 base Transformer 模型参数，bpe 对单语设置大小为 5k，分别在改进前后的 Transformer 模型上训练 20 轮，测试对比改进后模型对训练的影响。

2）BPE 对比实验

由于本实验提供的 iwslt'14 de-en 训练集较小，使用标准参数可能会造成过拟合，因此按照不同的 BPE 词典大小，分别设置模型的参数，并在改进后的 Transformer 模型上训练 20 轮，对不同 BPE 词典大小进行对比实验。

3）beam search 测试效果

在前两个实验中，已经对模型进行了性能上的测试，而为了使实验结果更直观，因此本处使用自己写的 beam search 函数在模型上进行测试。

六、实验结果及分析

1）Transformer 改进前后对比实验

将参考文献[1]中的基准 Transformer 模型与论文[2]中改进后的 Transformer 模型在同一数据集和相同训练集上进行对比实验，其训练过程中在验证集上的收敛速度如图 6.1 所示。通过 20 轮的训练，改进前模型在验证集上的 accuracy 收敛到 50 左右，而改进后模型则收敛到 66 左右。从图 6.1 中可以看出改进后模型

收敛速度有巨大提升，但依据参考文献[2]所述，其改进方法对模型的最终收敛程度提升不会很大，仅影响训练时的收敛速度，因此可以推断本实验中对原模型的 warn-up 等参数设置不够完善，导致其收敛程度远远低于改进后模型。

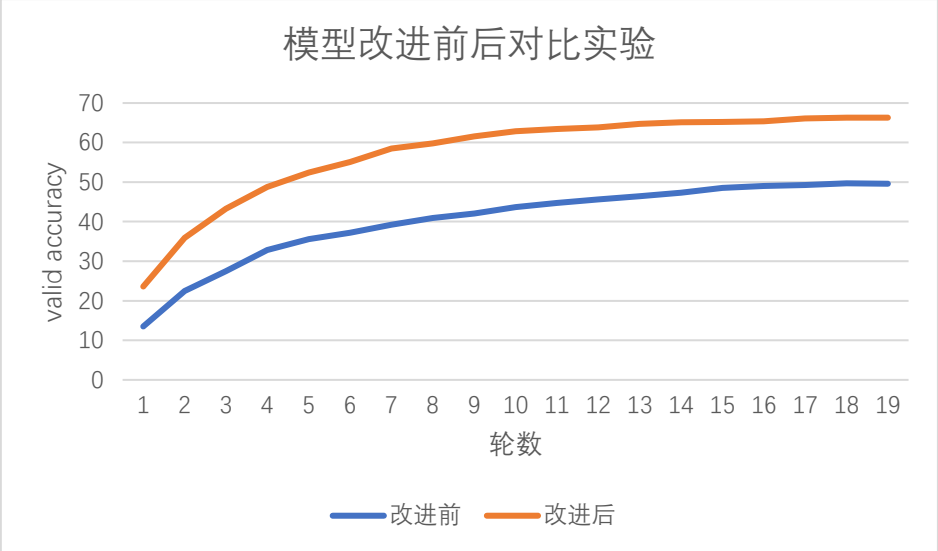


图 6.1 模型改进前后对比实验结果

2）BPE 对比实验

在上一实验中发现，如果将 Transformer 模型设置为 6 层，8 头，隐藏维度为 512，全连接层维度为 2048，则生成的模型文件为 200 多 MB，远远超过训练数据文件的 50MB。因此尝试通过减小 BPE 词典大小并适当调整模型参数的方式来缩减模型文件大小。将 BPE 单语设置为 2k 时，将 Transformer 模型设置为 3 层，4 头，隐藏维度为 256，全连接维度为 1024，此时生产的模型文件仅 20 多 MB（以下简称 mini 模型）。并且在该参数下，与标准模型参数、BPE 单语大小设置为 5K 的基准实验组进行对比的训练效果如图 6.2 所示。可以发现即使将 BPE 设置为 2k，减小模型参数，模型依旧可以在验证集上收敛到接近 70 的 accuracy，收敛程度和速度上较基准对照组有细微提升。

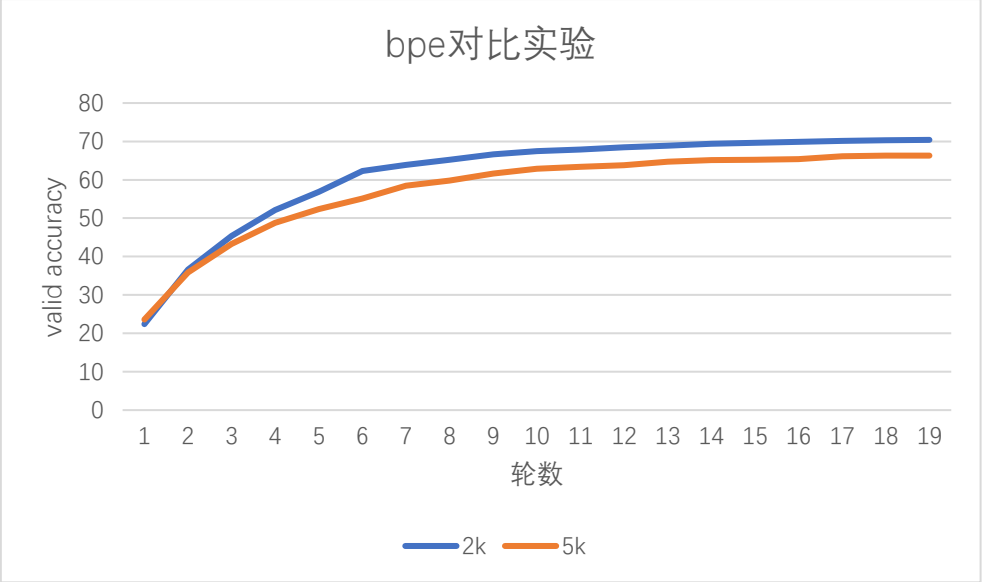


图 6.2 bpe 对比实验结果

同时，在测试集上对测试集进行 BLEU 得分计算，使用标准参数以及 BPE 大小为 5k 的基准模型在 BLEU-4 上得分为 32.61。而使用缩减模型及 BPE 大小设置为 2k 的 mini 模型在 BLEU-4 上的得分为 33.07，与基准模型几乎一致，这证明了使用缩减参数后的 mini 模型在在 iwsl'14 de-en 翻译任务上依旧表现很好。

3）beam search 测试效果

beam search 的束搜索宽度设置为 3，需用 3 条相同的测试语句，mini 模型的生成效果如下表 6-1 所示，bpe 大小为 5k 的基准模型生产效果如下表 6-2 所示。

表 6-1 mini 模型生成效果

测试用例 1	
德文输入	selbst wenn sie sich jetzt erschießen würden , hätten sie emissionen .
参考译文	even if you were to shoot yourself now , you 'd still have emissions .
生成 1	even if you shoot , you would have emissions .
生成 2	even if they shoot , they would have emissions .
生成 3	even if you shoot yourself , you would have emissions .
测试用例 2	
德文输入	es gibt kein futter .

参考译文	there is no feed .
生成 1	there 's no food .
生成 2	there is no food .
生成 3	there 's no feed .
测试用例 3	
德文输入	sie können eintauchen in ein bestimmtes jahr .
参考译文	you can drill into a particular year .
生成 1	you can dive into a certain year .
生成 2	you can dive in a certain year .
生成 3	you can dive into a particular year .

表 6-2 基准模型生成效果

测试用例 1	
德文输入	selbst wenn sie sich jetzt erschießen würden , hätten sie emissionen .
参考译文	even if you were to shoot yourself now , you 'd still have emissions .
生成 1	even if you shoot yourself , you would have emissions .
生成 2	even if you shoot themselves , you would have emissions .
生成 3	even if you shoot yourself , they would have emissions .
测试用例 2	
德文输入	es gibt kein futter .
参考译文	there is no feed .
生成 1	there 's no food .
生成 2	there is no food .
生成 3	there isn 't a food .
测试用例 3	
德文输入	sie können eintauchen in ein bestimmtes jahr .
参考译文	you can drill into a particular year .

生成 1	you can dive into one particular year .
生成 2	you can dive into a particular year .
生成 3	you can dive into a certain year .

从以上两个表格的内容可以看出 mini 模型与基准模型在该数据集上的表现均较为优秀，但 mini 模型的存储性能和计算速度要远远高于基准模型。

七、心得体会

在本次实验中，我收获很大。在之前的学习中，一直没有注意 Transformer 模型的训练细节，这也导致了我在本次实验中遇到了很多坑。

首先是模型的初始化，我之前阅读《Attention is all you need》论文时，没注意到参数的初始化，因此一直导致模型在一开始很难收敛，训练速度也非常慢。为此，我在 github 上查询了别人写法的 Transformer 模型，一开始我以为是我的注意力计算时出了错，因此我又重新 debug 了遍注意力机制部分的执行步骤，并且手动推导来检查代码是否有误，但这里并没有检查到问题，别人写的 Transformer 模型在注意力上的计算也是相同的方法。后来经过仔细排查后发现是模型忘记对参数进行初始化，我以为 pytorch 会自动初始化参数，然而事实是自动初始化的参数是随机的，并不服从正太分布或者均匀分布，导致模型很难收敛。发现这个问题后，我又测试了普通的正太分布初始化和推荐使用的 xavier_uniform，发现 xavier 的方式收敛速度更快，而普通的正太初始化则几乎对模型收敛速度影响不大。

其次是模型的结构，由于我参照的是最初版本的 Transformer 模型[1]，其虽然使用了 warm-up，但是我在实验过程中发现收敛到 accuracy 为 50 时就不再提升。通过与别人的代码对比，发现别人的 layer norm 层的位置与我的不太一样，经过查询发现，这种结构设置是来源参考文献[2]，改成这种结构后模型在测试集上的 accuracy 能收敛到 70。

此外，也尝试了使用自己写的 BPE 算法和 beam search 算法来训练和测试模型。并且通过将 BPE 词典大小设置为 2k，我发现 Transformer 模型的参数在 iwslt'14 de-en 数据集上可以设置的更小，是的模型训练速度、存储大小等方面

都获得极大提升。

最后，通过本次实验，我收获很大，也深刻的体会到了自己的不足，发现自己的文献阅读量太少，要是有更大的文献储备，一开始的两个问题都不会遇到；同时我也明白了理论结合实际的重要性：之前在阅读论文时，一直对注意力机制的计算不太清楚，而通过本次实验，我通过手动计算了注意力机制和 debug 相结合的方式，对注意力机制有了更清晰的认识，也对 Transformer 有了更加深入的了解。

八、参考文献

[1] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[C]. Advances In Neural Information Processing Systems. 2017: 5998-6008.

[2] Xiong R , Yang Y , He D , et al. On Layer Normalization in the Transformer Architecture[J]. 2020.