# 2021-2022学年东北大学

# 《语言分析与机器翻译》大作业报告

**姓 名** 胡义民

**专 业**  计算机科学与技术

**学 号**  2101730

**学 院**  计算机科学与工程学院

**任课教师** 肖桐、朱靖波

**2021年 12月 01日**

### 目 录

[一、 Transformer 3](#_Toc20626)

[二、 Dropout 4](#_Toc15984)

[三、 Base v.s. Tiny 5](#_Toc27350)

[四、 Pre-Norm v.s. Post-Norm 6](#_Toc20366)

[五、 Ensemble 7](#_Toc16348)

[六、 Knowledge Distillation 9](#_Toc16134)

### **Transformer**

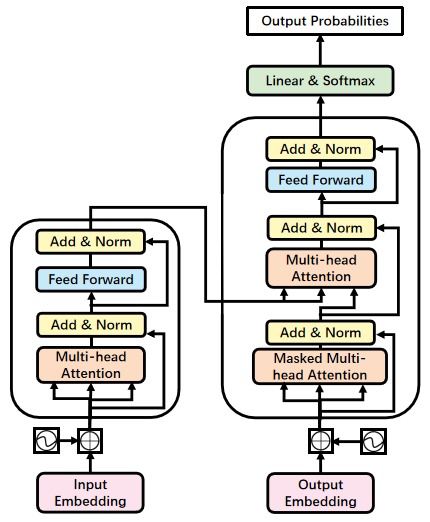


图1.1 Transformer模型结构

图1.1展示了Transformer的模型结构。模型的编码器由若干层堆叠而成，对于输入的向量序列，编码器层会输出同样大小的向量序列，每层又由若干个不同的模块组成，其中包括：

（1）多头自注意力子层（Multi-head Attention Sub­layer）：使用多头自注意力机制对输入序列许学习，从而得到新的表示；

（2）前馈神经网络子层（Feed Forward Sub­layer）：使用全连接层的前馈神经网络将输入序列进行进一步的变换；

（3）残差连接（Add）：对于自注意力子层和前馈神经网络子层，都有一个

从输入直接到输出的额外连接，也就是一个跨子层的直连。残差连接可以使深层网络的信息传递更为有效；

（4）层标准化（Layer Normalization）：自注意力子层和前馈神经网络子层进行最终输出之前，会对输出的向量进行层标准化，规范结果向量取值范围，这样易于后面进一步的处理。

以上的操作就构成了Transformer网络中的一层。编码器模块对于输入的执行顺序为：多头子注意力层、残差连接、层标准化、前馈神经网络子层、残差连接、再到层标准化。编码器与解码端类似，仅仅经过多头自注意力子层后引入了一个编码-解码多头交叉注意力子层，用于帮助模型使用源语言句子的表示信息生成目标语言不同位置的表示。两注意力子层的结构完全相同，但输入不相同。对于编码器的多头自注意力子层来说，其三个输入query、key、value都相同，而在解码器的编码-解码交叉注意力子层中，key和value是相同的，即为解码端输出的上下文向量，而query是解码器每个位置的表示。

### Dropout

在网络模型结构较为复杂时，会导致过度拟合训练数据，即在训练数据上损失与困惑度都很小，而在测试数据上损失与困惑度都升高。作为训练神经网络的一种trick选择，Dropout可以有效地缓解过拟合的发生，达到正则化的效果。它随机地删除不同的隐藏层节点，相当于在减小网络参数的同时训练不同的网络（如图2.1），并对多个不同的模型结果取平均，同时也减小了神经元之间复杂的共适应关系。Transformer在词嵌入与位置编码、残差连接、注意力操作与前馈神经网络等四个位置都引入了Dropout操作。

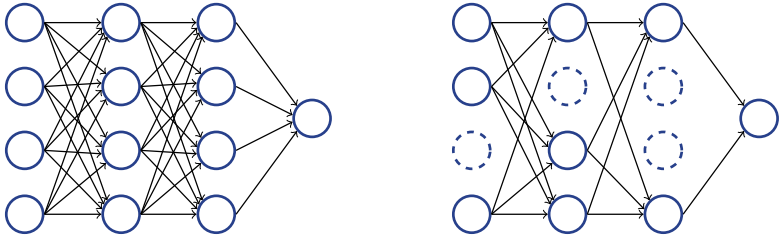


图2.1 使用Dropout前后的神经网络状态变化[1]

在给定的IWSLT14德英数据集上，基于Base模型结构，测试了设定不同比例Dropout的模型在训练了100轮后的性能：

表2.1 不同比例Dropout的模型性能比较

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 实验组号 | Dropout | BLEU | Best Valid ppl | Best Valid Epoch |
| 1 | 0.1 | 32.27 | 5.61 | 48 |
| 2 | 0.2 | 34.36 | 5.02 | 69 |
| **3** | **0.3** | **35.43** | **4.76** | **89** |
| 4 | 0.4 | 35.32 | 4.72 | 98 |
| 5 | 0.5 | 34.85 | 4.85 | 98 |
| 6 | 0.6 | 33.70 | 5.32 | 100 |
| 7 | 0.7 | 31.02 | 6.47 | 100 |
| 8 | 0.8 | 25.52 | 11.07 | 98 |
| 9 | 0.9 | 0.13 | 134.19 | 100 |

可以看到，随着dropout比例的增大，训练时保留的神经元比例过低，模型的收敛速度会减慢，表中当dropout增大到一定比例时，模型在训练100轮后完全没有收敛；其次，dropout比例过小时并不能充分缓解过拟合的发生，如dropout设定为0.1时，在训练50轮左右，模型还是会过拟合于训练数据，验证集上的困惑度会不降反升。

### Base v.s. Tiny

在标准的Transformer模型结构的解码器端，通常由六个结构相同的层堆叠而成，又由于当前时刻的预测输出对之前时刻的依赖关系，这种自回归性会大大降低解码计算的并行性，进而导致模型的自回归解码相当耗时，一种解决方法是减少解码器的层数，以获得模型性能与解码速度的折中。

在给定的IWSLT14德英数据集上，分别测试了Base与Tiny模型在训练了150轮后的性能：

表3.1 Base与Tiny模型性能比较

|  |  |  |  |
| --- | --- | --- | --- |
| 模型名称 | 解码器层数 | BLEU | Tokens/s |
| Base | 6 | 35.63 | 3003 |
| Tiny | 1 | 34.21 | 4698 |

由表3.1中的实验结果，在将解码器层数由6层直接减少到一层后，模型的性能其实并没有大幅度地下降，且解码的速度实现了进半的增长。

### Pre-Norm v.s. Post-Norm

标准的Transformer中，编码器与解码器通常会由6层标准结构堆叠而成，这些层又包含多个子层，因而即使是标准Transformer实际上也是一个相对较深的网络，再加上许多复杂的变换与计算，整个网络的信息传递会十分复杂，在这样的网络中进行反向传播容易引起梯度的消失与爆炸。为了保障模型的顺利训练，一种解决方法就是引入残差连接和层标准化。

残差连接会将前面所有层的输出加到一起：

(5.1)

其中，代表第层网络的输入向量，是子层运算，但这样会导致不同层结果间差别较大，导致训练不稳定或训练时间太长。层标准化操作可以缓解这种情况：

(5.2)

使用输入数据的均值，方差，对样本进行标准化，其中和是可学习的参数。

Transformer中的层标准化有两种，分别为前作与后作方式，后作方式的层标准化在残差连接之后进行，而前作方式会在子层输入前进行层标准化操作。



（a）后标准化 （b）前标准化

图5.1 层标准化的两种不同方式[1]

在给定的IWSLT14德英数据集上，基于Base模型结构，测试了后作与前作两种层标准化方式的模型在训练了150轮后的性能：

表4.1 Pre-Norm与Post-Norm模型性能比较

|  |  |  |
| --- | --- | --- |
| 模型名称 | 层标准化方式 | BLEU |
| Base | 后作 | 35.63 |
| Pre | 前作 | 34.53 |

由于Base与Pre模型的层数并不足够多，也就是模型不是特别深，采用前作的层标准化性能要略低于后作，但如果模型的层数加深，前作的层标准化方式将在后作无法有效进行训练的同时实现较优的性能。

### **Ensemble**

Ensemble是系统融合技术的一种，其本质是模型局部预测的融合。对于给定的模型，平均融合他们解码器输出的概率分布，相当于一种典型的线性插值模型，如图5.1所示。由于模型之间的互补性，多个模型的插值能够有效地降低经验错误率，通过集成不同模型的预测上限Oracle，有可能提高集成结果模型的预测上限，以实现模型性能的提升。

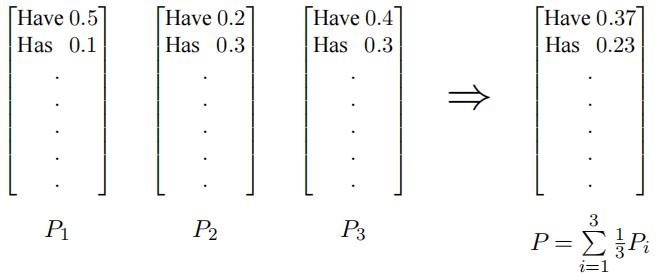


图5.1 Ensemble示例[1]

基于上述基于Base、Tiny与Pre的实验，使用ensemble方法集成了包含性能最好的5个epoch的模型预测结果，即通过不同时刻保存的模型得到更具鲁棒性的单模型，并测试了其性能，如表5.1所示：

表5.1 上述实验Ensemble模型性能比较

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 模型名称 | Ensemble | Epoch | Dropout | 解码器层数 | 层标准化方式 | BLEU | BLEU Before |
| Base | 5 | 54 | 0.1 | 6 | 后作 | 33.89 | 32.27 |
| Base | 5 | 100 | 0.1 | 6 | 后作 | 33.05 | 32.27 |
| Base | 5 | 100 | 0.2 | 6 | 后作 | 34.82 | 34.36 |
| Base | 5 | 100 | 0.3 | 6 | 后作 | 35.63 | 35.43 |
| Base | 5 | 100 | 0.4 | 6 | 后作 | 35.50 | 35.32 |
| Base | 5 | 100 | 0.5 | 6 | 后作 | 35.07 | 34.85 |
| Base | 5 | 100 | 0.6 | 6 | 后作 | 33.82 | 33.70 |
| Base | 5 | 100 | 0.7 | 6 | 后作 | 31.27 | 31.02 |
| Base | 5 | 100 | 0.8 | 6 | 后作 | 25.55 | 25.52 |
| Base | 5 | 100 | 0.9 | 6 | 后作 | 0.14 | 0.13 |
| Base | 5 | 150 | 0.3 | 6 | 后作 | 35.84 | 35.63 |
| Tiny | 5 | 150 | 0.3 | 1 | 后作 | 34.31 | 34.21 |
| Pre | 5 | 150 | 0.3 | 6 | 前作 | 34.91 | 34.53 |

上述实验中，可以看到经过ensemble方法集成后的单模型的性能都要优于某个单一时刻保存的性能最优的模型，也就是说即使是通过控制改变保存时刻这样简单又便捷的因素，集成的模型也有可能实现性能的提升，这更加印证了模型之间是存在着互补关系的。

当然，除了利用不同时刻保存的模型集成，还可以使用参数平均后的模型进行预测结果的融合，模型的多样性可以通过或异或同的模型结构与不同的随机种子实现，数据的多样性则可以通过不同的方法（如fine-tuning、bagging等）构造子模型来实现。[2]

### Knowledge Distillation

知识蒸馏是一种常见的知识迁移手段，由于模型中蕴含的知识规律是可迁移的，因此知识蒸馏可以将大模型的知识迁移到小模型当中，实现模型的压缩。又由于模型中所蕴含的知识规律比原始数据中的知识有更低的学习难度，因此能够更容易被学习到。知识蒸馏通常由词级与句子级知识蒸馏两种实现形式，词级知识蒸馏会最小化教师与学生模型输出分布之间的交叉熵，而句子级知识蒸馏会在输出序列的整体上进行预测概率的拟合。

分别以上述实验中得到的Base与Pre作为固定的教师模型，以Tiny与Dist（采用前作层标准化方法的Tiny）结构作为待训练的学生模型进行知识蒸馏，得到的实验结果如下表6.1所示：

表6.1 上述实验Ensemble模型性能比较

|  |  |  |  |
| --- | --- | --- | --- |
| 学生模型 | 教师模型 | Epoch | BLEU |
| Tiny | Base | 57 | 32.94 |
| Dist | Pre | 54 | 33.46 |

从上表6.1中的实验结果可以看出，作为学生模型的网络能够基本学习到教师模型中所蕴含的关于原始数据的知识，但性能其实还略显不足，这是因为知识蒸馏其实是对超参数甚至随机种子都较为敏感的技术，本实验的超参还值得进一步调优。

[1]肖桐,朱靖波. 机器翻译:基础与模型[M]. 电子工业出版社,2021.

[2]李北,王强,肖桐,姜雨帆,张哲旸,刘继强,张俐,于清. 面向神经机器翻译的集成学习方法分析[J]. 中文信息学报, 2019, 33(3): 42-51.