

**基于Transformer的情绪分类模型的初步认识与学习**

姓 名 ： 刘令文

学 号 ： 2201839

研 究 方 向 ： 医学信息计算

二○二二年秋季

1. 作业背景

本人从自动化本科来到计算机就读硕士研究生，对于各种深度学习模型尚无深刻认识与理解。有幸选修了肖桐老师的自然语言处理课程，使得我对Transformer模型有了一定的认识。此作业主要目的是熟悉Transformer的原理、基于Hugging Face的库实现简单的Transformer模型来进行情感分类任务。

2. Transformer模型

Transformer模型是由Google Brain在2017年时提出的一个注意力模型，也可以称作是一个特殊的编码-解码器模型，但同时又解决了传统编码-解码器模型的缺点。Transformer在使用注意力机制方面进行了创新，使得其能够注意到更重要的信息，并且能够避免长期依赖现象。图1是Transformer的模型架构，引用于《Attention Is All You Need》论文。

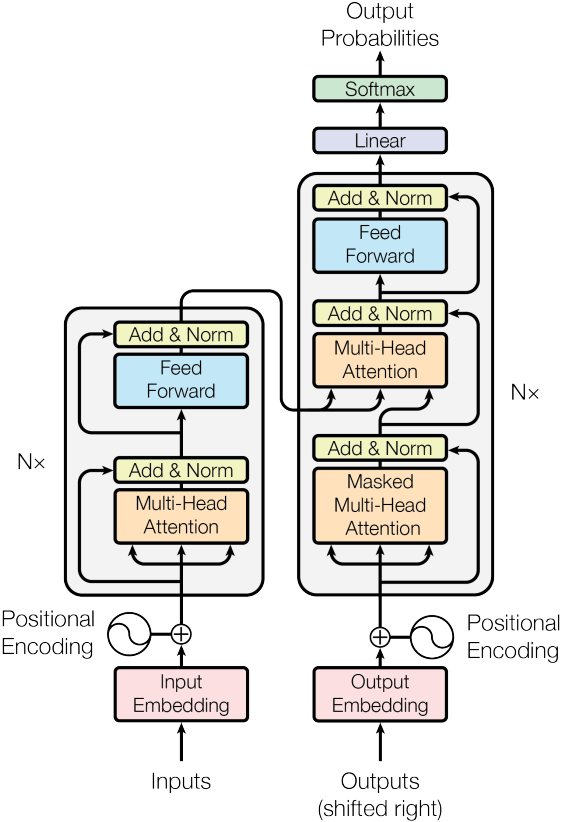


图1 Transformer模型结构（来自《Attention Is All You Need》论文）

BERT，则是Google提出的Transformer双向编码器，即Bidirectional Encoder Representations from Transformers。其优点是经过微调便可以用于很多种自然语言处理任务中，是该领域的一个转变性变革，BERT模型在NLP领域着实具有很大的影响力。其微调便可以使用的优点的背后也是巨大的预训练代价，BERT在大型语言库进行训练，才能够获得对语言原理的深入理解。

3. 实验设置

3.1 Hugging Face库

Hugging Face是一个位于纽约的公司，因为开源了一个关于Transformer模型的库而闻名于机器学习业界社区。目前拥有上十万多种已经预训练好的模型、以及上万种数据集。对于刚入门的初学者来说，Hugging Face是一个能够快速上手应用到Transformer模型的开源库，并且在调试好后一步一步地去熟悉模型。在本次作业中，使用Hugging Face的BERT预训练模型来实现对数据集内数据的预处理，即分词器，以及使用BERT模型作为模型的主体。除此之外，还使用Hugging Face进行数据集的获取及读取操作。

3.2 数据集

使用Hugging Face的数据集seamew/ChnSentiCorp，是一个中文的情感分类数据集。 在本次作业中，首先将该数据集克隆到本地，然后读取其中的csv文件。克隆方式采取Hugging Face上所介绍的：

1. git lfs install
2. git clone https:*//huggingface.co/datasets/seamew/ChnSentiCorp*

从而获得train.csv, dev.csv, test.csv三个文件。而对于数据的读取，则使用以下代码实现：

1. my\_dataset\_all = datasets.load\_dataset("csv", data\_files={
2. "train": "ChnSentiCorp\_csv/train.csv",
3. "validation": "ChnSentiCorp\_csv/dev.csv",
4. "test": "ChnSentiCorp\_csv/test.csv"},
5. cache\_dir='ChnSentiCorp\_csv')

最终获得的数据信息如下：

1. DatasetDict({
2. train: Dataset({
3. features: ['label', 'text'],
4. num\_rows: 9600
5. })
6. validation: Dataset({
7. features: ['label', 'text'],
8. num\_rows: 1200
9. })
10. test: Dataset({
11. features: ['label', 'text'],
12. num\_rows: 1200
13. })
14. })

3.3 模型代码主体

含有BERT和一个全连接层的模型主体：

1. **class** BERT\_Model(torch.nn.Module):
2. **def** \_\_init\_\_(self, BERT\_model):
3. super().\_\_init\_\_()
4. self.BERT = BERT\_model
5. self.fc1 = torch.nn.Linear(768, 256)
6. self.fc2 = torch.nn.Linear(256, 2)
8. **def** forward(self, ids, masks, tokenids):
9. output = self.BERT(ids, masks, token\_ids)
10. output = self.fc2(self.fc1(output[0][:, 0]))
11. output = output.softmax(dim=1)
12. **return** output

这其中，BERT\_model由Hugging Face获得：

1. transformers.BertModel.from\_pretrained('bert-base-uncased')

3.4 运行环境

下表为服务器环境：

表1 运行环境

|  |  |
| --- | --- |
| System | Ubuntu 20.04.4 LTS |
| CPU | Intel(R) Core (TM) i7-10700 CPU @ 2.90GHz |
| 内存 | 64G |
| GPU | NVIDIA A40 48G |

4. 实验结果

本次作业的实验结果如表2。从表格中可以看出，当只训练Linear模型来做分类时，可以得到62.6%的效果，然而当微调BERT模型时，其提升效果只比Linear模型高出0.5%，并不高。这里很大原因是由于我对Transformer的理解不够深刻，导致所写的代码或者模型架构存在错误，导致结果不理想。

表2 运行结果

|  |  |
| --- | --- |
| 模型 | ACC (%) |
| BERT(未参与训练)+Linear | 62.6 |
| BERT(参与训练)+Linear | 63.1 |

5. 总结

本次大作业从实践角度学习并尝试了一个简单的Transformer模型的代码实现，在这个过程中，我对Transformer模型有了更进一步的认识，受益匪浅。最后在代码实现的环节中，我通过结合Hugging Face库，实现了简单的模型，得到了62.6%和63.1%的结果，提升不大，其主要原因是我本人对Transformer模型的理解尚浅，因此还有很多的知识和盲区需要我去探索和学习。最后，感谢肖桐老师及其实验室的助教们的认真教学与细心辅导。