Rank1-Conversation_Emotion_Detection_Task

accuracy	macro-f1	recall
0.826	0.7544	0.719

基于预训练模型和时序预测模型的对话情感探测任务

1 摘要

2 方案

2.1 初步分析

题目形式：
要求根据对话历史，预测最后一句话的情绪。情绪包括 Happiness, Love, Sorrow, Fear, Disgust, None 对应着 1,2,3,4,5,6

1. 训练集中每段对话有n个句子，每个句子都有对应的情绪标签。
2. 测试集包括n个句子的对话以及前n-1个情绪标签。
3. 评测的排名指标是macro-f1值。

想到的三种做法：

1. 测试时只使用最后一句话。相当于简单的文本分类。
2. 测试时使用所有对话，以及前n-1个情绪标签。相当于先提取每句话的特征，再将这些句子特征通过时序预测模型进行预测。
3. 测试时只使用最后一句话，以及前n-1个情绪标签。相当于将任务分解为两个子任务，一个是对最后一句话进行文本分类，一个是使用标签进行时序预测。

方法1肯定是次优解，而2，3哪个更好不太好说，但是3在实现上更简单一些，所以我先尝试了1，在1的基础上尝试了3，没有尝试2

2.1 数据处理

为尝试方法1，3，将提供的数据集处理为文本分类和时序预测数据集

提供的数据集：

```
ID,Text,Labels
1,我就奇怪了　为啥你能拍得这么美呢 __eou__ 因为我做什么都认真，都诚心诚意！ __eou__ 好你诚心诚意！我谦虚
```

文本分类数据集：

1. 训练集：将对话按 __eou___ 符号分开，并将每句话和其对应的标签配对
2. 测试集：将对话按 __eou___ 符号分开，取最后一句话

时序预测数据集：

1. 训练集：将前n-1个标签(x)和最后一个标签(y)分开，对于数据集中对话长度不同的问题，使用padding方法，将x中长度不足的标签序列填充到x中最大的长度
2. 测试集：由于测试集中不含最后一个标签，直接将提供的标签padding

我曾尝试使用EDA（《EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks》）对文本分类数据做扩增，但在自己划分的测试集上效果反而下降了，我觉得有以下两点原因：

1. 通过观察标签分布，发现这是一个类别不均衡问题。 `Counter({6: 17549, 2: 6515, 1: 5651, 5: 4138, 3: 2534, 4: 432})` 。但我把所有标签对应的数据都进行了等比例的扩增，所以没有缓解类别不均衡。
2. EDA的扩增方式不够好，其中同义词替换是通过找词向量最相近的词，但是在很多时候并不合适，反而引入了噪声。我认为如果有一个语言模型进行辅助会更好一点。

2.2 模型

2.2.1 模型选择

在比赛中，我只使了用深度学习方法。对于**文本分类任务**，有以下几种方案：

1. 使用预训练的词/字向量+RNN/Text-CNN
2. 使用预训练模型+微调

由于1需要手动进行分词，而2一般有自带的tokenizer，省去了分词这一步，非常方便，而且效果一般来说远远超过传统的深度学习模型，所以我选择方法2。我使用的所有模型都是在transformers库中下载的，包括roberta-base-finetuned-dianping-chinese、chinese-electra-180g-large-discriminator 和 bert-base-chinese

对于**时间序列预测任务**，我采用GRU来学习前向关系，采用Transformer的Encoder来学习双向关系，以进行更好的预测。

2.2.2 参数设置

对于**预训练模型**，可以调节的参数有dropout概率，以及微调时更新全部参数还是只更新部分参数，更新哪部分参数。

我尝试了冻结Embedding层、冻结除分类层之外的所有层，以及不冻结参数，发现还是更新全部参数效果最好。

我在训练集中划分了更小一部分(8000)，采用网格搜索的方式，寻找每个模型的dropout概率，最后roberta设为0.05，bert设为0.1，electra设为0.3。

对于**时序预测模型**，因为这个预测问题非常简单，用特别小的模型就可以拟合。所以我使用了1层，模型维度为8，dropout概率为0.05的GRU模型；以及2层，模型维度为16，dropout概率为0.2，注意力头数为4的Transformer-Encoder。

2.2.3 训练方式

训练中采用的优化器均为AdamW，准则均为CrossEntropyLoss

微调预训练模型：
训练时的 learning rate 是最重要的超参。由于我的研究方向（机器翻译）中，学习率通常设为1e-3，所以我微调时想当然的把学习率设为1e-3，但是无论如何都不收敛。

别的地方都确认无误后，偶然发现把学习率改小似乎可以收敛，去重读BERT的论文，才发现文中已经给出微调时建议的学习率了（2e-5或5e-5）。

由于我还采用了先线性增加再线性衰减的学习率调整策略，所以还有一个 warmup steps 的超参，这个超参与 epoch num、batch size、learning rate 都有关系，所以一开始我想通过网格搜索的方式找到一个最优的组合，但是代价太大了，我就没有继续尝试。

根据多次尝试，我找到了一个适合所有模型的训练配置：在训练3轮，batch size为32的情况下，将learning rate设为2e-5，将warmup steps设为400。

训练时序预测模型：
时序预测模型与机器翻译任务相似，学习率设为2e-3即可收敛。

其他尝试：

1. 以上两类模型训练完后，我都曾尝试使用SGD进行继续训练，但似乎很难找到更好的点，所以后面就把这个步骤省略了。
2. 我还尝试了一种微调方法Child-Tuning（《Raise a Child in Large Language Model: Towards Effective and Generalizable Fine-tuning》）。其思想是更新梯度时只更新一部分参数的梯度，相当于对反向传播加了一个正则化。其中child-tuning-f是每次随机选择要更新的参数，child-tuning-d是先根据fisher信息计算一个集合（意为这个集合里的参数更无关紧要），以后只更新集合外的参数。不过两种方法我都进行了多次尝试，似乎没有明显的提升。

2.2.3 模型融合

单模型参数平均：
即把同一个模型的多个检查点的参数进行平均。对于每个模型，我使用10个不同的随机种子分别训练得到10个最优模型，但是发现平均模型参数的方法在我自己划分的测试集上效果会下降。我猜可能是每次训练时训练集和验证集都不一样，所以这10个模型的参数差异还是比较大。

预测结果融合：
即把多个模型最后输出的概率向量进行平均。我把上一步得到的10个模型的预测结果进行平均，发现效果有提升。但是考虑到训练代价，我没有进行单模型的多seeds融合。而是用这种方法**融合不同模型**，这样可以把两个单独的任务给结合起来，在预测时同时用到文本信息和历史的标签信息。

在测试集的效果来看，融合roberta+bert+electra三个模型，准确率仅在0.79左右，f-score刚刚0.7多一点；而再添加一个GRU，准确率直接提高到了0.823，f-score提高到0.73。而将GRU与Transformer-Encoder进行平均之后，准确率可以提升到0.828。

我还使用了一个很重要的trick，虽然上面的结果有很高的准确率，但是召回率总是很低，还不到0.7，这让我的f-score始终上不去。于是我重新训练了三个文本分类模型，这次我是按照验证集上recall最高的策略保存的模型（之前训练的模型都是按照验证集上f-score最高保存的）。这一次融合roberta+bert+electra三个模型可以达到0.724的recall。而时序预测模型保持不变，依旧使用f-score最高的GRU和Transformer-Encoder。最后融合5个模型之后，准确率依旧可以达到0.826，但召回率也提升到了0.719，此时f-score已经是7.544，达到了rank1。

3 总结