基于预训练模型和时序预测模型的对话情感探测任务

2101724 韩宇晨

1 摘要

针对对话情感探测任务，本文将其分为文本分类和时间序列预测两个子任务，分别采用预训练模型和时序预测模型进行拟合，并使用预测结果融合的方式将两种模型的预测结果结合在一起。在实验中，本文使用微调的 Roberta、Bert、Electra 三个模型，并与 GRU、Transformer-Encoder 相结合，得到了 0.7544 的 macro-f1 值，0.826 的准确率以及 0.719 的召回率，取得了 Rank1。

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。

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！我谦虚低调！咱都是优秀品格的人再赞一个 干杯 __eou__ 嗯嗯，咱俩都是最可爱的人！	2222

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，而方法 2，3 哪个更好不太好说，但是方法 3 在实现上更简单一些，所以我先尝试了方法 1，在方法 1 的基础上尝试了方法 3，没有尝试方法 2（方法 2 可能需要先用 Bert 来编码每一句，再用 LSTM 编码句子的序列关系；或者不区分句子，直接在词级别进行注意力平均，但是 Labels 的监督信号怎么用就成了问题）。

2.2 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

文本分类数据集：

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

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：

训练集：将前 n-1 个标签(x)和最后一个标签(y)分开，对于对话情绪标签序列长度不同的问题，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

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

由于方法 1 需要手动进行分词，而方法 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》）。其思想是反向传播更新参数时只更新一部分参数，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。对于每种预训练模型，我使用 10 个不同的随机种子分别微调得到 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。我猜可能因为 1）与机器翻译相比，这个任务的数据量还是偏少（10w-1000w VS 3w）。2）每次训练时由于随机划分（0.05），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 一些不足

在赛后观看了其他同学的 poster 后，发现自身有很多不足。

1）对数据不够重视。首先，我只想着用一个 tokenizer 搞定一切，没有对数

据做很好的清洗，可能数据中的一些 emojy 会导致 unk 的现象，这些我都没有去检查；另外，这是一个很严重的类别不平衡问题，我没有对这一点进行针对性的处理。我看到刘宝印同学使用了降采样，还有 k 折交叉验证的方法，我觉得他的结论对我非常有启发：某些折里 recall 非常低，经过观察发现这些折里发生了严重的类别失衡现象，通过模型平均，能使 recall 有一个很好的提升。这与我的猜测是一致的。最后，我总想着用端对端的方式直接提高得分，却忽视去观察数据中的一些现象。刘同学的 notebook 中做了很多图、表，对数据的分析可以说比较透彻，而我并没有做相关分析。

2）对网上现有工具的利用不够好。其实文本分类已经有现有的代码库，只需要把数据集处理成对应的形式就行了，但是我没有使用，所有代码都是自己实现的，花费了很多不必要的时间。

3）自身代码能力还需要提高：PyTorch 的 CrossEntropyLoss 内部其实已经包含了 Softmax 操作，所以只需要把模型输出的 logits 传进去即可。由于很久没用过，我想当然的把 logits 先经过 Softmax 再送给 CrossEntropyLoss，虽然也能收敛，但是效果会差一点。（dev accuracy 75 VS accuracy 72）。有一个现象就是一个 logits 经过多个嵌套的 Softmax 后，每个元素会趋于相同。知乎上也有过分析讨论：https://www.zhihu.com/question/403941326。

4 总结感悟

这是我第一次尝试文本分类比赛，一开始对 transformers 这个库的使用方式基本一头雾水，比如：加载预训练模型的时候，怎么改变模型的 dropout 概率？怎么把预训练模型的分类头改成我需要的类别数，改完之后怎么保存自己微调的模型，重新加载并在测试集上进行预测？不过这些问题都在查资料的过程中解决了，最终形成了自己的一套比赛框架。

这次比赛能幸运的成为 Rank1 的关键也是之前的一些积累，比如之前在机器翻译上尝试过的模型的集成方法；又比如这次使用的时序预测模型，其实是在我本科毕业设计 Simple NMT 中完成的，所以直接拿来就会用，也得到了很好的效果。

附录

比赛 Poster：

基于预训练模型和时序预测模型的对话情感探测任务

2101724 韩宇晨 **Code:** https://github.com/hannlp/Rank1-Conversation_Emotion_Detection_Task **Blog:** hannlp.github.io(Welcome ^ ^)

摘要

针对对话情感探测任务，本文将其分解为**文本分类**和**时间序列预测**两个子任务，分别采用**预训练模型**和**时序预测模型**进行拟合，并使用预测结果融合的方式将两类模型的预测结果结合在一起。在实验中，本文使用微调的Roberta、Bert、Electra三个模型，并与GRU、Transformer-Encoder相结合，得到了0.7544的f1值，0.826的accuracy以及0.719的recall。

分析数据，确定解法

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。**太难，×**
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。√

数据处理

文本分类数据集：按__eou__切开，与标签配对即可
时序预测数据集：将前n-1个标签(x)和最后一个标签(y)分开，对于标签序列长度不同的问题，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

何为padding？看图
（本科毕业论文里拷的）

模型选择与超参

model	p drop
Roberta	0.05
Bert	0.1
Electra	0.3

model	d model	p drop	n layers	n heads
GRU	8	0.05	1	\
Transformer-Enc	16	0.2	2	4

优化器均为AdamW，准则均为交叉熵损失，预训练模型学习率和warmup steps均为2e-5和400，时序预测模型学习率均为2e-3。

预测结果融合方法★

两类模型的输出都是6维向量，把多个模型最后的输出先过Softmax再平均即可。

实验

model ensemble	accuracy	macro-f1	recall
第一次训练 Roberta+Bert+Electra	0.788	0.705	0.69
+GRU+Transformer	**0.828**	0.735	0.69
第二次训练 Roberta+Bert+Electra	0.77	忘了	**0.724**
+GRU+Transformer	0.826	**0.7544**	0.719

其他尝试

1.数据扩增：EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks。尝试失败，可能没处理好数据不均衡问题
2.新的微调技术：Raise a Child in Large Language Model: Towards Effective and Generalizable Fine-tuning。尝试失败，怀疑科研

机器翻译大作业：

由于我的本科毕设就是基于 Transformer 的机器翻译系统 SimpleNMT，所以这次机器翻译大作业我交的是其他题目。不过我用 SimpleNMT 在 iwslt14.de-en 上跑了一遍，并写了相应的运行脚本 train.sh 和 evaluate.sh，结果在：

https://github.com/hannlp/SimpleNMT/tree/main/examples/iwslt14