多模态情感分析实验报告

实验环境

pandas==1.1.5

anaconda-client==1.7.2

anaconda-navigator==2.0.3

conda==4.11.0

huggingface-hub==0.16.4

ipykernel

ipython

jupyter==1.0.0

numpy

tokenizers==0.13.3

torch==2.0.1

torchaudio==2.0.2+cu117

torchdata==0.6.1

torchtext==0.15.2

torchvision==0.15.2+cu117

transformers==4.30.2

模型选择及理由

· 多模态融合模型可以结合图片和文本的特征，获取更全面、准确的情感信息，进而做出更准确的情感倾向判断。

· Bert模型：本多模态融合模型的文本特征提取是基于Bert模型。Bert是基于Transformer架构的预训练语言模型，能够同时考虑到单词前后的上下文信息，捕捉到丰富的语义特征。同时，由于BERT模型在预训练阶段能够包含大量信息，将其作为特征提取器，通过迁移学习将已学的语义信息传递给下游的情感分析任务，可以提高整个模型的准确率。

· Resnet模型：图片特征提取是基于Resnet模型。Resnet50是基于ImageNet数据集上的预训练图像模型，是深度学习网络，能够在深层网络中训练和优化，提取高级语义信息。Resnet模型通过残差学习、跨层连接，可以将原始特征和残差特征结合在一起，提高了图像特征的表达能力，可以提高图像情感分析的准确率。

实验步骤

数据载入及预处理

· 由于数据集是一个包含图片文本编号的文件夹，所以先定义了一个文本路径获取和图像路径获取的函数，通过路径加载训练集的图片与文本。同时为了便于训练，将情感标签替换为对应的0，1，2。

· 按照固定比例8：2划分训练集与验证集，每个数据集都包含图片、文本、标签。

· 文本处理：使用BertTokenizer的一个经过预训练的分词器和BertModel的预训练模型，通过自定义的text_preprocess函数进行文本处理。传入文本列表，遍历输入的每个文本，使用分词器对文本进行分词并进行padding和truncation操作，最后将结果转换为PyTorch的张量形式，以准备输入到BERT模型中进行后续的文本特征提取和处理任务。

· 图像处理：将图像也处理统一的格式，转化成张量格式。

· 定义数据集类：自定义Dataset用来存储图片路径、文本、标签和transform。

将图像数据、文本数据和标签组合在一起，便于在训练过程中按批次加载和处理。

构建模型

多模态融合模型的核心思想是学习融合的图片、文本特征，于是先分别定义了一个图片特征提取模型和文本特征提取模型。

· 文本特征提取模型。

调用了经过预训练的BERT模型作为文本特征提取器，在前向传播过程，输入分词后的文本和attention_mask（指示哪些元素是真实文本，哪些是填充元素的attention mask）作为参数，通过BERT模型对文本进行特征提取，返回提取到的文本特征表示。

· 图片特征提取模型

调用torchvision.models中一个经过预训练的ResNet-50模作为图像特征提取器，提取出图像的高级语义特征。定义前向传播函数，输入图像张量，通过ResNet-50模型提取图像的特征表示，并输出这些特征表示。

· 多模态融合模型

核心思想：根据option参数，实现在图像、文本或图文的选择

1.首先初始化前面定义的图像特征提取器、文本特征提取器

2.初始化三分类器，每个optipn下的分类器都由两个全连接层组成，输入图象时维度定义为1000，文本为768，图文为1768；随后通过ReLU激活函数，还添加了Dropout层防止过拟合，最后通过线性函数输出维度为3的分类。

3.前向传播。输入图像、分词后的文本和attention_mask。根据option参数的值来确定采用哪种模式进行前向传播：

- 如果option为0，只使用图像特征提取器对图像进行处理，并通过图像分类器进行分类。
- 如果option为1，只使用文本特征提取器对文本进行处理，并通过文本分类器进行分类。

- 如果option为其他值，同时使用图像特征提取器和文本特征提取器对图像和文本进行处理，然后将两者的特征表示进行拼接，并通过多模态融合分类器进行分类。

训练模型

1.定义训练函数：

初始化损失为0。使用train_loader循环迭代训练数据的每个批次，每批包含图像、输入文本、attention_mask和标签。先通过optimizer.zero_grad清空优化器的梯度缓存，随后将输入数据传递给模型进行前向传播，得到输出。将输出与目标比较，统计预测正确的总数，并通过通过损失函数计算损失并累加。

通过loss.backward进行反向传播，不断更新模型的参数。每批次训练结束后，计算平均损失值和准确率。

保存最优训练模型。

2.定义预测函数：

先创建一个空列表存储预测结果，遍历数据集，先将输入的文本数据进行压缩去除多余的维度，通过多模态情感分析模型对输入数据进行预测，得到预测输出。

3.具体训练过程：

1. 首先，初始化损失函数、训练批次数、训练轮数epoch与学习率lr，因为要考察lr参数对模型的影响，于是定义了一个学习率列表，每一轮初始化一个学习率为l的Adam优化器。
2. 遍历lr列表，每一轮化初始一个多模态融合模型，输入模型类型（文本、图片还是两者都）
3. 遍历每个epoch，调用训练函数训练模型，并返回训练损失和训练准确率，同时调用预测函数对验证集数据对进行预测，计算预测结果准确率。与最佳准确率比较，不断更新记录最佳准确率。
4. 每得到一个最佳准确率，保存当前模型，最后可以得到一个最优模型。

生成预测结果

读取验证集上表现最好的模型，对测试集数据进行预测，并写回文件。

实验结果（在训练集、验证集上的结果）

第一轮：lr设置为0.1，从第二个epoch开始模型不收敛，怀疑时lr过高

```
lr: [0.001, 0.01, 0.1], Epoch 1/10, Train Loss: 1.1068, Train Acc: 0.5956, Val Acc: 0.5962
lr: [0.001, 0.01, 0.1], Epoch 2/10, Train Loss: 1.0986, Train Acc: 0.5972, Val Acc: 0.5962
lr: [0.001, 0.01, 0.1], Epoch 3/10, Train Loss: 1.0986, Train Acc: 0.5972, Val Acc: 0.5962
lr: [0.001, 0.01, 0.1], Epoch 4/10, Train Loss: 1.0986, Train Acc: 0.5972, Val Acc: 0.5962
lr: [0.001, 0.01, 0.1], Epoch 5/10, Train Loss: 1.0986, Train Acc: 0.5972, Val Acc: 0.5962
lr: [0.001, 0.01, 0.1], Epoch 6/10, Train Loss: 1.0986, Train Acc: 0.5972, Val Acc: 0.5962
lr: [0.001, 0.01, 0.1], Epoch 7/10, Train Loss: 1.0986, Train Acc: 0.5972, Val Acc: 0.5962
lr: [0.001, 0.01, 0.1], Epoch 8/10, Train Loss: 1.0986, Train Acc: 0.5972, Val Acc: 0.5962
lr: [0.001, 0.01, 0.1], Epoch 9/10, Train Loss: 1.0986, Train Acc: 0.5972, Val Acc: 0.5962
lr: [0.001, 0.01, 0.1], Epoch 10/10, Train Loss: 1.0986, Train Acc: 0.5972, Val Acc: 0.5962
```

第二轮：lr设置为0.001，模型收敛效果依旧较差，在训练集和测试集上的准确率都保持不变。

```
training on  cpu
lr: 0.001, Epoch 1/10, Train Loss: 1.4802, Train Acc: 0.5739, Val Acc: 0.5968
lr: 0.001, Epoch 2/10, Train Loss: 0.4804, Train Acc: 0.5971, Val Acc: 0.5968
lr: 0.001, Epoch 3/10, Train Loss: 0.3080, Train Acc: 0.5971, Val Acc: 0.5968
lr: 0.001, Epoch 4/10, Train Loss: 0.2279, Train Acc: 0.5971, Val Acc: 0.5968
lr: 0.001, Epoch 5/10, Train Loss: 0.1815, Train Acc: 0.5971, Val Acc: 0.5968
lr: 0.001, Epoch 6/10, Train Loss: 0.1510, Train Acc: 0.5971, Val Acc: 0.5968
lr: 0.001, Epoch 7/10, Train Loss: 0.1293, Train Acc: 0.5971, Val Acc: 0.5968
lr: 0.001, Epoch 8/10, Train Loss: 0.1132, Train Acc: 0.5971, Val Acc: 0.5968
lr: 0.001, Epoch 9/10, Train Loss: 0.1006, Train Acc: 0.5971, Val Acc: 0.5968
lr: 0.001, Epoch 10/10, Train Loss: 0.0905, Train Acc: 0.5971, Val Acc: 0.5968
最佳准确率
0.5967540574282147
```

第三轮：lr设置为0.0001，模型收敛效果优化。

```
training on  cpu
lr: 1e-05, Epoch 1/10, Train Loss: 0.1476, Train Acc: 0.9565, Val Acc: 0.6017
lr: 1e-05, Epoch 2/10, Train Loss: 0.0740, Train Acc: 0.9594, Val Acc: 0.6017
lr: 1e-05, Epoch 3/10, Train Loss: 0.0533, Train Acc: 0.9562, Val Acc: 0.6017
lr: 1e-05, Epoch 4/10, Train Loss: 0.0330, Train Acc: 0.9565, Val Acc: 0.6030
lr: 1e-05, Epoch 5/10, Train Loss: 0.0238, Train Acc: 0.9594, Val Acc: 0.6017
lr: 1e-05, Epoch 6/10, Train Loss: 0.0213, Train Acc: 0.9597, Val Acc: 0.6017
lr: 1e-05, Epoch 7/10, Train Loss: 0.0195, Train Acc: 0.9612, Val Acc: 0.6017
lr: 1e-05, Epoch 8/10, Train Loss: 0.0172, Train Acc: 0.9603, Val Acc: 0.6017
lr: 1e-05, Epoch 9/10, Train Loss: 0.0151, Train Acc: 0.9565, Val Acc: 0.6017
lr: 1e-05, Epoch 10/10, Train Loss: 0.0123, Train Acc: 0.9641, Val Acc: 0.6017
最佳准确率
0.602996254681648
```

消融实验结果

只输入文本：

```
training on  cpu
lr: 0.01, Epoch 1/10, Train Loss: 2.1805, Train Acc: 0.4986, Val Acc: 0.5618
lr: 0.01, Epoch 2/10, Train Loss: 0.3848, Train Acc: 0.6777, Val Acc: 0.5880
lr: 0.01, Epoch 3/10, Train Loss: 0.1646, Train Acc: 0.8109, Val Acc: 0.6067
lr: 0.01, Epoch 4/10, Train Loss: 0.1113, Train Acc: 0.8450, Val Acc: 0.6005
lr: 0.01, Epoch 5/10, Train Loss: 0.0745, Train Acc: 0.8737, Val Acc: 0.6005
lr: 0.01, Epoch 6/10, Train Loss: 0.0603, Train Acc: 0.8984, Val Acc: 0.6080
lr: 0.01, Epoch 7/10, Train Loss: 0.0472, Train Acc: 0.9125, Val Acc: 0.5980
lr: 0.01, Epoch 8/10, Train Loss: 0.0425, Train Acc: 0.9137, Val Acc: 0.5955
lr: 0.01, Epoch 9/10, Train Loss: 0.0326, Train Acc: 0.9209, Val Acc: 0.6167
lr: 0.01, Epoch 10/10, Train Loss: 0.0386, Train Acc: 0.9212, Val Acc: 0.6055
```

只输入图片：

```
training on  cpu
lr: 0.01, Epoch 1/10, Train Loss: 0.9253, Train Acc: 0.5971, Val Acc: 0.5968
lr: 0.01, Epoch 2/10, Train Loss: 0.4560, Train Acc: 0.5971, Val Acc: 0.5968
lr: 0.01, Epoch 3/10, Train Loss: 0.3025, Train Acc: 0.5971, Val Acc: 0.5968
lr: 0.01, Epoch 4/10, Train Loss: 0.2265, Train Acc: 0.5971, Val Acc: 0.5968
lr: 0.01, Epoch 5/10, Train Loss: 0.1811, Train Acc: 0.5971, Val Acc: 0.5968
lr: 0.01, Epoch 6/10, Train Loss: 0.1509, Train Acc: 0.5971, Val Acc: 0.5968
lr: 0.01, Epoch 7/10, Train Loss: 0.1293, Train Acc: 0.5971, Val Acc: 0.5968
lr: 0.01, Epoch 8/10, Train Loss: 0.1131, Train Acc: 0.5971, Val Acc: 0.5968
lr: 0.01, Epoch 9/10, Train Loss: 0.1006, Train Acc: 0.5971, Val Acc: 0.5968
lr: 0.01, Epoch 10/10, Train Loss: 0.0905, Train Acc: 0.5971, Val Acc: 0.5968
最佳准确率
0.5967540574282147
```

实验结论

本实验的多模态融合模型在训练集、验证集和测试集上的表现都优于单文本或图片。

但是整体上模型表现较一般，最高准确率只有60%，目前能力受限无法优化。

问题与解决

1.计算交叉熵时报错RuntimeError: "nll_loss_forward_reduce_cuda_kernel_2d_index" not implemented for 'Int'

解决：加上.long()转换数据格式，改为l = loss(y_hat, y.long())

2.终端安装transfomer库失败，报错ERROR: Could not find a version that satisfies the requirement transformers==2.0.0//ERROR: No matching distribution found for transformers==2.0.0

解决：关掉梯子就可以解决下载问题

3.ValueError: check_hostname requires server_hostname

解决：关掉梯子

4.TypeError: 'BertTokenizer' object is not callable

原因：transforms版本低于3.0.0不能直接使用

解决：pip install transformers==3.4.0

5.TypeError: Descriptors cannot not be created directly.If this call came from a _pb2.py file, your generated code is out of date and must be regenerated with protoc >= 3.19.0.

解决：先安装了 protobuf==3.19.0，这个错误解决，但又报了个新错：Couldn't build proto file into descriptor pool: duplicate file name sentencepiece_model.proto，经查阅是因为包的版本过高，后又加载了低版本，解决。

6.OSError: Can't load weights for 'bert-base-multilingual-cased'. Make sure that:- 'bert-base-multilingual-cased' is a correct model identifier listed on 'https://huggingface.co/models'- or

'bert-base-multilingual-cased' is the correct path to a directory containing a file named one of pytorch_model.bin, tf_model.h5, model.ckpt.

解决：卸载transformer并重装，输入命令：python -m pip install --upgrade pip -i https://pypi.douban.com/simple --user

7.导入完成后，加载预训练bert模型和tokenizer有时会报错（heck_hostname requires server_hostname），有时又能正常运行。

解决：关掉梯子，解决

8.Input type (int) and bias type (float) should be the same。resnet网络的精度和数据精度不一致

解决：一开始未对图片数据进行张量化处理导致报错，现在手动对输入数据进行张量化torch.Tensor(X)

9.AttributeError: 'bool' object has no attribute 'sum'

解决：原先的predicted==labels是布尔型，不能直接sum()，于是将predicted和labels转化成数组格式，逐个判断是否相等，sum累加。