华东师范大学数据与工程学院实验报告

课程名称：当代人工智能	年级：2020级	上机实践成绩：
指导教师：李翔	姓名：何雨晴	学号：10205501458
上机实践名称：多模态情感分析	上机实践日期：	2023.6
上机实践编号：5	组号：	上机实践时间：

本项目的代码已经上传至github代码仓库Miaheeee/AI_lab5: 多模态情感分析 (github.com)

一、实验任务

- 给定配对的文本和图像，预测对应的情感标签。
- 三分类任务：positive, neutral, negative。

  例：输入 '??? #stunned #sunglasses #gafas #gafasdesol \n'

输出：Positive

二、数据预处理

观察数据集中图片的文字说明后发现，语句中存在一些对情感分类无用的字符串，例如：@xxx、#、http...等等，考虑到这些文字可能会干扰模型预测，所以在预处理阶段对文字说明进行过滤，筛掉无意义文字。

```
    for i in range(len(descriptions)):
        des = descriptions[i]
        word_list = des.replace("#", "").split(" ")
        words_result = []
        for word in word_list:
            if len(word) < 1:
                continue
            elif (len(word)>=4 and 'http' in word) or word[0]=='@':
                continue
            else:
                words_result.append(word)
        descriptions[i] = " ".join(words_result)
```

train.txt共有4000条数据，按照8：2的比例划分为训练集和测试集。为了方便模型预测之后计算准确率，将情绪标签按分类映射为0，1，2，`emo_tag = {"neutral": 0, "negative": 1, "positive": 2}`，并且查看每种情绪的占比如下。

```
1 :  0.2983245811452863
0 :  0.10477619404851213
2 :  0.5968992248062015
```

三、模型构建

3.1 BERT+RESNET

本项目是完成多模态情感分析任务，故需要分别提取文本的特征以及图片的特征，将二者特征融合之后再进行情绪的分类。对于文本进行情感分析，就是需要得到整个文本的上下文的综合信息，想到Bert模型在处理文本信息时，内部采用自注意力机制，最后对每个字输出一个向量，并且在句子开头添加了一个特殊符号'CLS'，该符号对应的向量中包含了整个句子的综合信息。所以，进行情绪分类的文本特征考虑采用'CLS'的向量信息。对于图片的特征提取，考虑使用在ImageNet上预训练过的Resnet，之所以选择Resnet是因为越深的网络提取的特征越抽象，越具有语义信息，Resnet中有残差结构，该结构使得训练更深的网络时缓解梯度消失或者梯度爆炸的问题。

3.1.1 简单拼接

首先考虑将分别通过bert和resnet提取到的文字特征和图片特征进行简单的拼接之后，通过一个全连接层得到在三个类别上的分数，得分最高的类别即为最终预测的类别。

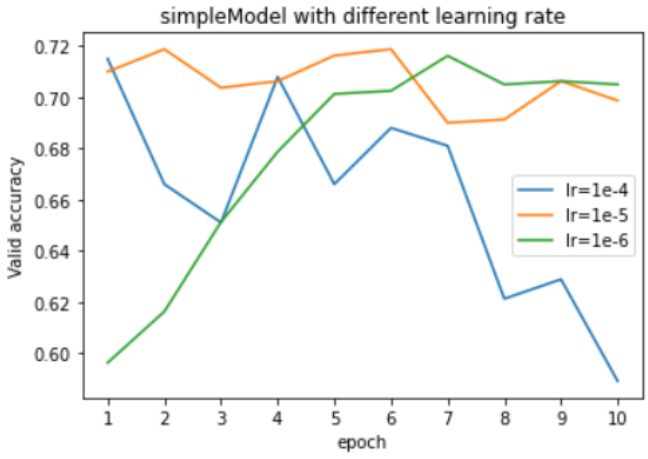
```
class simpleModel(nn.Module):
    def __init__(self):
        super().__init__()
        self.txt_model = BertModel.from_pretrained('bert-base-uncased')
        self.img_model = torchvision.models.resnet18(pretrained=True)
        self.t_linear = nn.Linear(768, 128)
        self.i_linear = nn.Linear(1000, 128)
        self.fc = nn.Linear(256, 3)
        self.relu = nn.ReLU()

    def forward(self, input_ids, attention_mask, image):
        img_out = self.img_model(image)
        img_out = self.i_linear(img_out)
```

```
        img_out = self.relu(img_out)
        txt_out = self.txt_model(input_ids=input_ids,
attention_mask=attention_mask)
        txt_out = txt_out.last_hidden_state[:,0,:]
        txt_out.view(txt_out.shape[0], -1)
        txt_out = self.t_linear(txt_out)
        txt_out = self.relu(txt_out)
        last_out = torch.cat((txt_out, img_out), dim=-1)
        last_out = self.fc(last_out)
        return last_out
```

以下是学学习率设置为1e-5时的训练过程：

```
bert_resnet_simple
Train Epoch: 1, Train_Loss: 0.6856, Train Accuracy: 0.6500, Valid Accuracy: 0.7100
Train Epoch: 2, Train_Loss: 0.6532, Train Accuracy: 0.8053, Valid Accuracy: 0.7188
Train Epoch: 3, Train_Loss: 0.2189, Train Accuracy: 0.9237, Valid Accuracy: 0.7037
Train Epoch: 4, Train_Loss: 0.3424, Train Accuracy: 0.9613, Valid Accuracy: 0.7063
Train Epoch: 5, Train_Loss: 0.0208, Train Accuracy: 0.9775, Valid Accuracy: 0.7163
Train Epoch: 6, Train_Loss: 0.0110, Train Accuracy: 0.9856, Valid Accuracy: 0.7188
Train Epoch: 7, Train_Loss: 0.2323, Train Accuracy: 0.9909, Valid Accuracy: 0.6900
Train Epoch: 8, Train_Loss: 0.0180, Train Accuracy: 0.9909, Valid Accuracy: 0.6913
Train Epoch: 9, Train_Loss: 0.0041, Train Accuracy: 0.9928, Valid Accuracy: 0.7063
Train Epoch: 10, Train_Loss: 0.0037, Train Accuracy: 0.9938, Valid Accuracy: 0.6987
```

调整学习率，观察在验证集上的准确率变化，可以看出在初始学习率为1e-6时，该模型的收敛效果较好。



3.1.2 加权融合

考虑到有些情况下图片特征对情感分类更重要，有些情况下文字特征对情感分类更重要，故采用动态加权的方式融合二者特征。实现思路是使用文本特征和图片特征和一个向量进行内积之后求得各自的权重，之后进行加权求和。

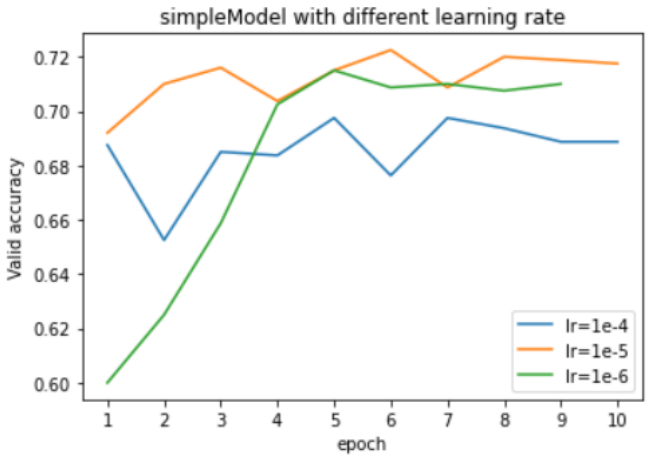
```
def forward(self, input_ids, attention_mask, image):
        img_out = self.img_model(image)
        img_out = self.i_linear(img_out)
        img_out = self.relu(img_out)
        img_weight = self.img_q(img_out)
        txt_out = self.txt_model(input_ids=input_ids,
attention_mask=attention_mask)
        txt_out = txt_out.last_hidden_state[:,0,:]
        txt_out.view(txt_out.shape[0], -1)
        txt_out = self.t_linear(txt_out)
```

```
        txt_out = self.relu(txt_out)
        txt_weight = self.txt_q(txt_out)
        last_out = img_weight * img_out + txt_weight * txt_out
        last_out = self.fc(last_out)
        return last_out
```

以下是学学习率设置为1e-5时的训练过程：

```
Train Epoch: 1, Train_Loss: 0.5329, Train Accuracy: 0.6272, Valid Accuracy: 0.6987
Train Epoch: 2, Train_Loss: 0.1789, Train Accuracy: 0.7825, Valid Accuracy: 0.7200
Train Epoch: 3, Train_Loss: 0.2814, Train Accuracy: 0.8984, Valid Accuracy: 0.6813
Train Epoch: 4, Train_Loss: 0.0293, Train Accuracy: 0.9569, Valid Accuracy: 0.7188
Train Epoch: 5, Train_Loss: 0.0546, Train Accuracy: 0.9784, Valid Accuracy: 0.7238
Train Epoch: 6, Train_Loss: 0.0169, Train Accuracy: 0.9872, Valid Accuracy: 0.7188
Train Epoch: 7, Train_Loss: 0.0039, Train Accuracy: 0.9888, Valid Accuracy: 0.7175
Train Epoch: 8, Train_Loss: 0.0017, Train Accuracy: 0.9916, Valid Accuracy: 0.6987
Train Epoch: 9, Train_Loss: 0.0035, Train Accuracy: 0.9925, Valid Accuracy: 0.7238
Train Epoch: 10, Train_Loss: 0.0007, Train Accuracy: 0.9925, Valid Accuracy: 0.7137
```

调整学习率，观察在验证集上的准确率变化，可以看出在初始学习率为1e-5时，在验证集上的准确率最高能达到0.724。



3.2 BERT+DENSENET

由于本项目使用的数据集只有4000条训练数据，规模较小，故考虑采用Densenet进行提取图片的特征。因为小数据集的时候容易产生过拟合，但是Densenet能够很好的解决过拟合的问题，相比于一般神经网络的分类器直接依赖于网络最后一层（复杂度最高）的特征，DenseNet 可以综合利用浅层复杂度低的特征，因而更容易得到一个光滑的具有更好泛化性能的决策函数。

以下是学学习率设置为1e-5时的训练过程：

```
Train Epoch: 1, Train_Loss: 0.8572, Train Accuracy: 0.6444, Valid Accuracy: 0.6887
Train Epoch: 2, Train_Loss: 0.5608, Train Accuracy: 0.8122, Valid Accuracy: 0.7113
Train Epoch: 3, Train_Loss: 0.0697, Train Accuracy: 0.9381, Valid Accuracy: 0.7312
Train Epoch: 4, Train_Loss: 0.0045, Train Accuracy: 0.9778, Valid Accuracy: 0.7063
Train Epoch: 5, Train_Loss: 0.0226, Train Accuracy: 0.9891, Valid Accuracy: 0.7063
Train Epoch: 6, Train_Loss: 0.0021, Train Accuracy: 0.9912, Valid Accuracy: 0.7250
Train Epoch: 7, Train_Loss: 0.0013, Train Accuracy: 0.9931, Valid Accuracy: 0.7137
Train Epoch: 8, Train_Loss: 0.0015, Train Accuracy: 0.9928, Valid Accuracy: 0.7188
Train Epoch: 9, Train_Loss: 0.0033, Train Accuracy: 0.9922, Valid Accuracy: 0.7175
Train Epoch: 10, Train_Loss: 0.0475, Train Accuracy: 0.9934, Valid Accuracy: 0.7075
```

3.3 消融实验

仅输入图片，初始学习率设置为1e-5，训练结果如下，在验证集上最高准确率为0.64。

```
Train Epoch: 1, Train_Loss: 0.9700, Train Accuracy: 0.5906, Valid Accuracy: 0.6125
Train Epoch: 2, Train_Loss: 0.6145, Train Accuracy: 0.6562, Valid Accuracy: 0.6238
Train Epoch: 3, Train_Loss: 0.5443, Train Accuracy: 0.7119, Valid Accuracy: 0.6438
Train Epoch: 4, Train_Loss: 0.5976, Train Accuracy: 0.7662, Valid Accuracy: 0.6288
Train Epoch: 5, Train_Loss: 0.3353, Train Accuracy: 0.8237, Valid Accuracy: 0.6338
Train Epoch: 6, Train_Loss: 0.1773, Train Accuracy: 0.8875, Valid Accuracy: 0.6275
Train Epoch: 7, Train_Loss: 0.8600, Train Accuracy: 0.9300, Valid Accuracy: 0.6262
Train Epoch: 8, Train_Loss: 0.1220, Train Accuracy: 0.9656, Valid Accuracy: 0.6275
Train Epoch: 9, Train_Loss: 0.1772, Train Accuracy: 0.9772, Valid Accuracy: 0.6050
Train Epoch: 10, Train_Loss: 0.0303, Train Accuracy: 0.9834, Valid Accuracy: 0.6200
```

仅输入说明文字，初始学习率设置为1e-5，训练结果如下，在验证集上最高准确率为0.70。

```
Train Epoch: 1, Train_Loss: 0.6966, Train Accuracy: 0.6434, Valid Accuracy: 0.6937
Train Epoch: 2, Train_Loss: 0.7231, Train Accuracy: 0.7841, Valid Accuracy: 0.7013
Train Epoch: 3, Train_Loss: 0.1661, Train Accuracy: 0.8922, Valid Accuracy: 0.6900
Train Epoch: 4, Train_Loss: 0.0868, Train Accuracy: 0.9522, Valid Accuracy: 0.6963
Train Epoch: 5, Train_Loss: 0.1610, Train Accuracy: 0.9678, Valid Accuracy: 0.6850
Train Epoch: 6, Train_Loss: 0.0095, Train Accuracy: 0.9759, Valid Accuracy: 0.6937
Train Epoch: 7, Train_Loss: 0.0531, Train Accuracy: 0.9772, Valid Accuracy: 0.6937
Train Epoch: 8, Train_Loss: 0.1369, Train Accuracy: 0.9766, Valid Accuracy: 0.6937
Train Epoch: 9, Train_Loss: 0.0038, Train Accuracy: 0.9775, Valid Accuracy: 0.6963
Train Epoch: 10, Train_Loss: 0.0031, Train Accuracy: 0.9781, Valid Accuracy: 0.6775
```

由实验结果可以看出，仅使用图片在验证集上的准确率最高只能达到0.64，模型预测效果很不好。

四、遇到的bug

bug1：读入图片描述时，出现无法解码的问题。

```
main.py:32: DeprecationWarning: ANTIALIAS is deprecated and will be removed in Pillow 10 (2023-07-01). Use Resampling.LANCZOS
instead.
  img = img.resize((224,224),Image.ANTIALIAS)
Traceback (most recent call last):
  File "main.py", line 35, in <module>
    des = f.read()
UnicodeDecodeError: 'gbk' codec can't decode byte 0xac in position 75: illegal multibyte sequence
```

解决方案：根据代码给出的提示信息，可以看出错误的意思是：Unicode的解码（Decode）出现错误了，以gbk编码的方式去解码（该字符串变成Unicode），但是此处通过gbk的方式，却无法解码（can't decode）."illegal multibyte sequence"的意思是非法的多字节序列，也就是说无法解码了。出现这样的错误，可能是要处理的字符串本身不是gbk编码，却是以gbk编码去解码。比如，字符串本身是utf-8的，但用gbk去解码，必然出错。故采用更宽范围的编码进行解码，`with open('./data/' + str(guid) + '.txt', encoding='gb18030') as f:`，问题即可解决。

bug2：运行模型时，出现以下问题。

```
Traceback (most recent call last):
  File "main.py", line 152, in <module>
    main()
  File "main.py", line 137, in main
    model = simpleModel.to(device)
  File "/root/miniconda3/lib/python3.8/site-packages/torch/nn/modules/module.py", line 907, in to
    return self._apply(convert)
AttributeError: 'torch.device' object has no attribute '_apply'
```

解决方案：该问题出现的原因是构建的模型类没有实例化就使用了，进行实例化之后问题解决。

bug3：加载hugging face上的预训练模型时，出现无法连接http://huggingface.co的问题。

```
OSError: We couldn't connect to 'https://huggingface.co' to load this file, couldn't find it in the cached files and it looks
like bert-base-multilingual-cased is not the path to a directory containing a file named config.json.
Checkout your internet connection or see how to run the library in offline mode at 'https://huggingface.co/docs/transformers/
installation#offline-mode'.
```

解决方案：在`from_pretrained`函数里调用mirror选项，如`token = BertTokenizer.from_pretrained(pre_trained_model, mirror='bfsu')`，问题解决。

五、结果对比

模型	在训练集合上的准确率
bert+resnet简单拼接	0.719
bert+resnet权重融合	0.724
bert+densenet权重融合	0.732

六、实验心得

本项目实现了多模态情感分析，主要是使用Bert模型提取文字特征，使用Resnet/Densenet提取图片特征，以及对比运用了不同的特征融合方式。从实验结果来看，使用Bert和Densenet在训练集上的准确率能达到0.99，但在验证集上的准确率最高只能达到0.732，考虑原因可能是数据集的规模小，使得模型能拟合训练数据，而不具备良好的泛化能力。此外，本项目还进行了消融实验，对比了仅输入图片和仅输入文字的效果，由实验结果可以看出，文字说明对情绪判断有更大的帮助，仅使用图片无法较好的判断出情绪，考虑该情况出现的原因可能是使用的Resnet/Densenet都是使用Imagenet进行训练的，原目标是为了提取物品进行分类，故不能直接适用本项目想实现的情绪分类任务。