报告

GitHub地址

https://github.com/SunsetHe/ai-exp5

遇到的BUG

第一个BUG

在原本的代码思路中，是在一个python文件中进行

读入数据——定义CNN模型并将jpg转为特征向量——定义bert模型并将text转为特征向量

但是测试发现，如果第二部分的代码和第三部分的代码在同一处地方，会导致jpg的特征向量变得很奇怪

下面是没有第三部分时，第二部分的代码的某个特征向量

```
Train Image Vectors Length: tensor([0.1081, 0.2034, 0.1785,  ..., 0.2074, 0.1911,
0.1969], device='cuda:0')
```

下面是有第三部分时，第二部分的代码的某个特征向量（这里只放了简单的，如果查看全部数值，向量全部都是0 。

```
Train Image Vectors Length: tensor([0., 0., 0.,  ..., 0., 0., 0.], device='cuda:0')
```

可以发现有第三部分时，特征向量中含有大量的0，而正常情况下应该是没有那么多0的，改了很久也没有解决，最终只能靠编写好几个py程序分开运行解决此问题。

第二个BUG

划分测试集时，原本使用的代码是

```
train_data, valid_data = train_test_split(train_data_all, test_size=0.2,
random_state=42)
```

后来因为要编写好几个py程序，所以得把第一次划分之后的数据保存到本地，此时我换了种划分方式

```
train_data = train_data_all.copy()[:3200]
valid_data = train_data_all.copy()[3200:]
```

但是发现如果使用这种方式划分出来的训练集和测试集，所有的jpg的特征向量都是0，最后也没有解决这个问题，只能换回原来的划分方式，原因可能是[3200:]这种方式是浅复制什么的（我也不是很了解）

模型设计

个人的模型设计思路就是：读入数据——定义CNN模型并将jpg转为特征向量——定义bert模型并将text转为特征向量——将两个向量进行拼接——将拼接后的向量送入模型进行训练。

设计模型的思路是在CSDN和知乎上搜了点文章看了，但是大多数也没看懂，就看出来好像基本都是这么做的。

模型训练选择的是transformer模型，因为比较热门。

CNN模型

```
class SimpleCNN(torch.nn.Module):
    def __init__(self):
        super(SimpleCNN, self).__init__()
        self.conv1 = torch.nn.Conv2d(in_channels=3, out_channels=2, kernel_size=3,
stride=1, padding=1)
        self.relu1 = torch.nn.ReLU()
        self.pool1 = torch.nn.MaxPool2d(kernel_size=2, stride=2)
        self.conv2 = torch.nn.Conv2d(in_channels=2, out_channels=2, kernel_size=3,
stride=1, padding=1)
        self.relu2 = torch.nn.ReLU()
        self.pool2 = torch.nn.MaxPool2d(kernel_size=2, stride=2)
        self.flatten = torch.nn.Flatten()
```

bert模型

```
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained('bert-base-uncased').to(device)
```

transformer

```
class MultiModalTransformer(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(MultiModalTransformer, self).__init__()
        self.linear1 = nn.Linear(input_size, hidden_size)
        self.relu = nn.ReLU()
        self.linear2 = nn.Linear(hidden_size, output_size)
```

多模态融合模型在验证集上的结果

当jpg的特征向量长度为1024，text的特征向量的形状为[1, 768]时，

超参数lr：0.001

epoch：10

训练结果如下

```
Epoch 1/10, Training Loss: 0.8639, Training Accuracy: 60.69%, Validation Accuracy:
63.25%
Epoch 2/10, Training Loss: 0.7621, Training Accuracy: 67.28%, Validation Accuracy:
63.50%
Epoch 3/10, Training Loss: 0.7230, Training Accuracy: 68.62%, Validation Accuracy:
67.00%
Epoch 4/10, Training Loss: 0.6996, Training Accuracy: 70.09%, Validation Accuracy:
68.00%
Epoch 5/10, Training Loss: 0.6788, Training Accuracy: 71.81%, Validation Accuracy:
68.00%
Epoch 6/10, Training Loss: 0.6625, Training Accuracy: 72.31%, Validation Accuracy:
65.62%
Epoch 7/10, Training Loss: 0.6760, Training Accuracy: 71.25%, Validation Accuracy:
67.12%
Epoch 8/10, Training Loss: 0.6600, Training Accuracy: 72.12%, Validation Accuracy:
67.50%
Epoch 9/10, Training Loss: 0.6360, Training Accuracy: 73.06%, Validation Accuracy:
66.50%
Epoch 10/10, Training Loss: 0.6373, Training Accuracy: 73.22%, Validation Accuracy:
65.62%
```

可以发现当来到了5~6个epoch时已经趋于收敛，再往后就过拟合了，因此答案预测时选择的是6个epoch的模型。

消融实验结果

我的做法是

如果是只用jpg训练，则将text的向量变为同样形状的全部0向量，把这个和jpg向量拼接，再按之前的代码训练，下面是10个epoch的训练结果

```
Epoch 1/10, Training Loss: 0.9264, Training Accuracy: 59.94%, Validation Accuracy:
58.75%
Epoch 2/10, Training Loss: 0.9187, Training Accuracy: 59.94%, Validation Accuracy:
58.75%
Epoch 3/10, Training Loss: 0.9219, Training Accuracy: 59.94%, Validation Accuracy:
58.75%
Epoch 4/10, Training Loss: 0.9154, Training Accuracy: 59.94%, Validation Accuracy:
58.75%
Epoch 5/10, Training Loss: 0.9155, Training Accuracy: 59.94%, Validation Accuracy:
58.75%
Epoch 6/10, Training Loss: 0.9133, Training Accuracy: 59.94%, Validation Accuracy:
58.75%
Epoch 7/10, Training Loss: 0.9137, Training Accuracy: 59.94%, Validation Accuracy:
58.75%
Epoch 8/10, Training Loss: 0.9118, Training Accuracy: 59.94%, Validation Accuracy:
58.75%
Epoch 9/10, Training Loss: 0.9105, Training Accuracy: 59.94%, Validation Accuracy:
58.75%
Epoch 10/10, Training Loss: 0.9107, Training Accuracy: 59.94%, Validation Accuracy:
58.75%
```

可以发现训练几乎不收敛，训练集准确率和验证集准确率都比原始版本更低

下面是只使用text训练的数据

```
Epoch 1/10, Training Loss: 0.8707, Training Accuracy: 59.97%, Validation Accuracy:
64.12%
Epoch 2/10, Training Loss: 0.7604, Training Accuracy: 66.88%, Validation Accuracy:
65.00%
Epoch 3/10, Training Loss: 0.7178, Training Accuracy: 68.91%, Validation Accuracy:
67.75%
Epoch 4/10, Training Loss: 0.6959, Training Accuracy: 70.50%, Validation Accuracy:
67.38%
Epoch 5/10, Training Loss: 0.6794, Training Accuracy: 71.62%, Validation Accuracy:
68.62%
Epoch 6/10, Training Loss: 0.6754, Training Accuracy: 71.03%, Validation Accuracy:
66.88%
Epoch 7/10, Training Loss: 0.6479, Training Accuracy: 73.34%, Validation Accuracy:
65.62%
Epoch 8/10, Training Loss: 0.6402, Training Accuracy: 73.44%, Validation Accuracy:
66.62%
Epoch 9/10, Training Loss: 0.6309, Training Accuracy: 73.25%, Validation Accuracy:
69.12%
Epoch 10/10, Training Loss: 0.6208, Training Accuracy: 73.69%, Validation Accuracy:
67.50%
```

可以发现取得了跟原始版本几乎一样的成绩，我认为这可能是因为text的向量中含有更多有用信息导致的。

后续探索

因为在看的文章中提到了要控制不同数据之间的比例，所以我猜测jpg的特征向量长度和text的特征向量的形状占比的不同会导致训练结果的不同，下面是一些探索

下面是只对jpg的特征向量长度进行变化后的训练结果（因为CNN模型好调一点）

jpg的特征向量长度：3136,使用两种数据训练

```
Epoch 1/10, Training Loss: 0.8747, Training Accuracy: 59.66%, Validation Accuracy: 
61.62%
Epoch 2/10, Training Loss: 0.7950, Training Accuracy: 64.69%, Validation Accuracy: 
64.25%
Epoch 3/10, Training Loss: 0.7420, Training Accuracy: 68.66%, Validation Accuracy: 
64.62%
Epoch 4/10, Training Loss: 0.7224, Training Accuracy: 68.38%, Validation Accuracy: 
65.50%
Epoch 5/10, Training Loss: 0.6981, Training Accuracy: 69.47%, Validation Accuracy: 
65.75%
Epoch 6/10, Training Loss: 0.6829, Training Accuracy: 70.81%, Validation Accuracy: 
65.25%
Epoch 7/10, Training Loss: 0.6747, Training Accuracy: 71.19%, Validation Accuracy: 
65.50%
Epoch 8/10, Training Loss: 0.6611, Training Accuracy: 72.09%, Validation Accuracy: 
65.62%
Epoch 9/10, Training Loss: 0.6659, Training Accuracy: 72.12%, Validation Accuracy: 
65.50%
Epoch 10/10, Training Loss: 0.6516, Training Accuracy: 72.31%, Validation Accuracy: 
66.50%
```

只使用jpg训练

```
Epoch 1/10, Training Loss: 0.9220, Training Accuracy: 58.97%, Validation Accuracy:
58.75%
Epoch 2/10, Training Loss: 0.9082, Training Accuracy: 59.94%, Validation Accuracy:
58.75%
Epoch 3/10, Training Loss: 0.9069, Training Accuracy: 59.94%, Validation Accuracy:
58.75%
Epoch 4/10, Training Loss: 0.9074, Training Accuracy: 59.94%, Validation Accuracy:
58.75%
Epoch 5/10, Training Loss: 0.9091, Training Accuracy: 59.94%, Validation Accuracy:
58.75%
Epoch 6/10, Training Loss: 0.9102, Training Accuracy: 59.94%, Validation Accuracy:
58.75%
Epoch 7/10, Training Loss: 0.9075, Training Accuracy: 59.94%, Validation Accuracy:
58.75%
Epoch 8/10, Training Loss: 0.9051, Training Accuracy: 59.94%, Validation Accuracy:
58.75%
Epoch 9/10, Training Loss: 0.9053, Training Accuracy: 59.94%, Validation Accuracy:
58.75%
Epoch 10/10, Training Loss: 0.9063, Training Accuracy: 59.94%, Validation Accuracy:
58.75%
```

跟长度为1024时一样

只使用text训练

```
Epoch 1/10, Training Loss: 0.8549, Training Accuracy: 60.59%, Validation Accuracy:
61.75%
Epoch 2/10, Training Loss: 0.7543, Training Accuracy: 67.47%, Validation Accuracy:
63.62%
Epoch 3/10, Training Loss: 0.7063, Training Accuracy: 69.47%, Validation Accuracy:
64.88%
Epoch 4/10, Training Loss: 0.6858, Training Accuracy: 70.78%, Validation Accuracy:
65.12%
Epoch 5/10, Training Loss: 0.6769, Training Accuracy: 71.78%, Validation Accuracy:
66.25%
Epoch 6/10, Training Loss: 0.6539, Training Accuracy: 72.06%, Validation Accuracy:
66.75%
Epoch 7/10, Training Loss: 0.6442, Training Accuracy: 72.62%, Validation Accuracy:
66.00%
Epoch 8/10, Training Loss: 0.6432, Training Accuracy: 73.62%, Validation Accuracy:
68.62%
Epoch 9/10, Training Loss: 0.6304, Training Accuracy: 73.50%, Validation Accuracy:
67.75%
Epoch 10/10, Training Loss: 0.6213, Training Accuracy: 73.88%, Validation Accuracy:
66.38%
```

比使用两种数据训练稍差一些

---

jpg的特征向量长度：6272,使用两种数据训练

```
Epoch 1/10, Training Loss: 0.8845, Training Accuracy: 59.69%, Validation Accuracy: 
58.75%
Epoch 2/10, Training Loss: 0.8253, Training Accuracy: 63.19%, Validation Accuracy: 
64.38%
Epoch 3/10, Training Loss: 0.7651, Training Accuracy: 67.09%, Validation Accuracy: 
63.50%
Epoch 4/10, Training Loss: 0.7282, Training Accuracy: 68.62%, Validation Accuracy: 
65.75%
Epoch 5/10, Training Loss: 0.7107, Training Accuracy: 70.09%, Validation Accuracy: 
65.38%
Epoch 6/10, Training Loss: 0.7238, Training Accuracy: 68.94%, Validation Accuracy: 
64.50%
Epoch 7/10, Training Loss: 0.7146, Training Accuracy: 69.66%, Validation Accuracy: 
65.25%
Epoch 8/10, Training Loss: 0.7014, Training Accuracy: 70.50%, Validation Accuracy: 
65.50%
Epoch 9/10, Training Loss: 0.6849, Training Accuracy: 71.53%, Validation Accuracy: 
66.00%
Epoch 10/10, Training Loss: 0.6742, Training Accuracy: 72.25%, Validation Accuracy: 
65.50%
```

比原来稍差，原来的版本验证集准确率在后面的epoch会高两三个点。

只使用jpg训练

```
Epoch 1/10, Training Loss: 0.9260, Training Accuracy: 58.81%, Validation Accuracy:
58.75%
Epoch 2/10, Training Loss: 0.9111, Training Accuracy: 59.94%, Validation Accuracy:
58.75%
Epoch 3/10, Training Loss: 0.9104, Training Accuracy: 59.94%, Validation Accuracy:
58.75%
Epoch 4/10, Training Loss: 0.9184, Training Accuracy: 59.94%, Validation Accuracy:
58.75%
Epoch 5/10, Training Loss: 0.9100, Training Accuracy: 59.94%, Validation Accuracy:
58.75%
Epoch 6/10, Training Loss: 0.9089, Training Accuracy: 59.94%, Validation Accuracy:
58.75%
Epoch 7/10, Training Loss: 0.9067, Training Accuracy: 59.94%, Validation Accuracy:
58.75%
Epoch 8/10, Training Loss: 0.9079, Training Accuracy: 59.94%, Validation Accuracy:
58.75%
Epoch 9/10, Training Loss: 0.9078, Training Accuracy: 59.94%, Validation Accuracy:
58.75%
Epoch 10/10, Training Loss: 0.9069, Training Accuracy: 59.94%, Validation Accuracy:
58.75%
```

比原来差，而且跟更少的只使用jpg的一样

只使用text训练

```
Epoch 1/10, Training Loss: 0.8780, Training Accuracy: 58.97%, Validation Accuracy:
59.88%
Epoch 2/10, Training Loss: 0.7706, Training Accuracy: 66.09%, Validation Accuracy:
64.50%
Epoch 3/10, Training Loss: 0.7250, Training Accuracy: 68.81%, Validation Accuracy:
63.75%
Epoch 4/10, Training Loss: 0.6969, Training Accuracy: 69.72%, Validation Accuracy:
64.88%
Epoch 5/10, Training Loss: 0.6777, Training Accuracy: 71.44%, Validation Accuracy:
66.75%
Epoch 6/10, Training Loss: 0.6637, Training Accuracy: 72.56%, Validation Accuracy:
68.25%
Epoch 7/10, Training Loss: 0.6548, Training Accuracy: 72.84%, Validation Accuracy:
66.38%
Epoch 8/10, Training Loss: 0.6421, Training Accuracy: 73.19%, Validation Accuracy:
66.38%
Epoch 9/10, Training Loss: 0.6372, Training Accuracy: 73.41%, Validation Accuracy:
66.75%
Epoch 10/10, Training Loss: 0.6247, Training Accuracy: 73.53%, Validation Accuracy:
67.88%
```

跟短一点的没什么不一样。