

华东师范大学数据科学与工程学院实验报告

课程名称：当代人工智能

年级：

上机实践成绩：

指导教师：

姓名：易志鹏

学号：10215501426

上机实践名称：

上机实践日期：

上机实践编号：

组号：

上机实践时间：

一、实验任务：

给定配对的文本和图像，构建并训练多模态模型，预测对应的情感标签。

github 地址：[desrch/Multimodel_sentiment_analysis \(github.com\)](https://github.com/desrch/Multimodel_sentiment_analysis)

二、模型结构 (Multi_model.py)

1. 决策层融合 (decision_avg)

模型中嵌入了一个 bert 和一个 resnet 模型用于提取文本和图像的特征，随后将各自提取的特征放入线性分类器（2 层的 MLP）中得到 2 个决策（2 个预测概率），将两个决策结果相加，最后经 softmax 归一化得到最终的预测概率。

特点：决策层的平均融合使得模型具有较强的鲁棒性，即使某一模态数据缺失或噪声较大，模型仍能保持相对稳定的性能

2. 特征层融合 (attention_cat)

模型通过 bert 和 resnet 模型分别提取出文本和图像的特征，然后将这两个特征矩阵拼接在一起，经 Transformer Encoder 处理得到联合特征，最后将联合特征放入 MLP 分类器中得到最终的预测概率。

特点：因为不同模态输入的数据可能包含冗余或互补的信息，Transformer Encoder 中的多头注意力机制可以分析不同模态特征中的信息的重要性，进而一定程度上缓解信息冗余。同时多头注意力也能更好地捕捉不同模态信息中潜在的依赖关系，获取更全面的信息，进一步提升模型性能。

三、实验结果

1. 多模态模型结果

相关参数均采用代码中的默认参数配置，详见 multi_train.py

decision_avg:

验证集准确率：65.625%

```

args = parse.parse_args()
return args

1 个用法
def get_train_id_tag(train_file):
args_parse()

运行 multi_train (1) x
:
训练集损失: 0.8825189662910998
验证集损失: 0.951213747933507
验证集准确率: 0.65125
Epoch :19
训练集损失: 0.8784348418749869
验证集损失: 0.948031372949481
验证集准确率: 0.6525
Epoch :20
训练集损失: 0.873619807548821
验证集损失: 0.9491489145159722
验证集准确率: 0.65625
最佳模型准确率: 0.65625

```

attention_cat:

验证集准确率: 70.625%

```

1 import argparse
2 import os
3 import chardet
4 import numpy as np
5 import torch
6 from torch.nn.utils.rnn import pad_sequence
7 from transformers import AutoModel, AutoTokenizer
8 from torchvision.models import resnet101

if __name__ == '__main__': 2

运行 multi_train (1) x
:
训练集损失: 0.6110013725794852
验证集损失: 0.8536173544824124
验证集准确率: 0.69125
Epoch :20
训练集损失: 0.6109621905907988
验证集损失: 0.843886192664504
验证集准确率: 0.70625
最佳模型准确率: 0.70625
进程已结束, 退出代码为 0

```

2. 消融实验结果

采用 **attention_cat** 作为实验模型,一个模态输入正常数据, 另一模态输入空字符串或空白图片, 训练参数均采用默认参数配置。

text only :

准确率: 68.75%

```

59         self.fuse_dropout = torch.nn.Dropout(0.3)
60
61         self.fc_out = torch.nn.Linear(2 * hidden_dim, num_labels)
62         self.softmax = torch.nn.Softmax(dim=1)
63
64     def __init__(self):
65         super().__init__()
66         self.embedding = torch.nn.LSTMEmbedder(EMBED_DIM, HIDDEN_DIM)
67         self.encoder = torch.nn.LSTM(HIDDEN_DIM, HIDDEN_DIM, 1, True)
68         self.decoder = torch.nn.LSTM(HIDDEN_DIM, HIDDEN_DIM, 1, True)
69         self.fuse_dropout = torch.nn.Dropout(0.3)
70         self.fc_out = torch.nn.Linear(2 * hidden_dim, num_labels)
71         self.softmax = torch.nn.Softmax(dim=1)
72
73     def forward(self, x):
74         x = self.embedding(x)
75         x, _ = self.encoder(x)
76         x = self.decoder(x)
77         x = self.fuse_dropout(x)
78         x = self.fc_out(x)
79         x = self.softmax(x)
80         return x
81
82     def train(self, train_loader, val_loader):
83         for epoch in range(1, num_epochs + 1):
84             train_loss, train_acc = train_step(self, train_loader)
85             val_loss, val_acc = val_step(self, val_loader)
86             print(f'Epoch :{epoch} \n 训练集损失: {train_loss} \n 验证集损失: {val_loss} \n 验证集准确率: {val_acc}')
87             if val_acc > best_val_acc:
88                 best_val_acc = val_acc
89                 best_model_path = os.path.join('nice_models', f'best_model.pt')
90                 torch.save(self.state_dict(), best_model_path)
91
92 if __name__ == '__main__':
93     main()

```

运行 only_train

```

Epoch :18
训练集损失: 0.8427543276548386
验证集损失: 1.0322605186700822
验证集准确率 : 0.67
Epoch :19
训练集损失: 0.8364921236597002
验证集损失: 1.026188036352396
验证集准确率 : 0.67625
Epoch :20
训练集损失: 0.8315821276977658
验证集损失: 1.0218603652715683
验证集准确率 : 0.68625
最佳模型准确率: 0.6875

```

image only :

准确率: 66.375%

```

253     optimizer = torch.optim.AdamW(params, lr=args.lr)
254     num_epochs = args.epoch
255     total_steps = len(train_data_loader) * num_epochs
256     # scheduler = get_linear_schedule_with_warmup(optimizer, num_warmup_steps=0, num_training_steps=total_steps)
257     loss_fn = torch.nn.CrossEntropyLoss()
258     model.to(device)
259
260     best_accuracy = 0
261     val_accuracy = []
262
263     if __name__ == '__main__':
264         main()

```

运行 only_train

```

验证集准确率 : 0.65875
Epoch :20
训练集损失: 0.8364877933822572
验证集损失: 0.9541210868954658
验证集准确率 : 0.66375
最佳模型准确率: 0.66875

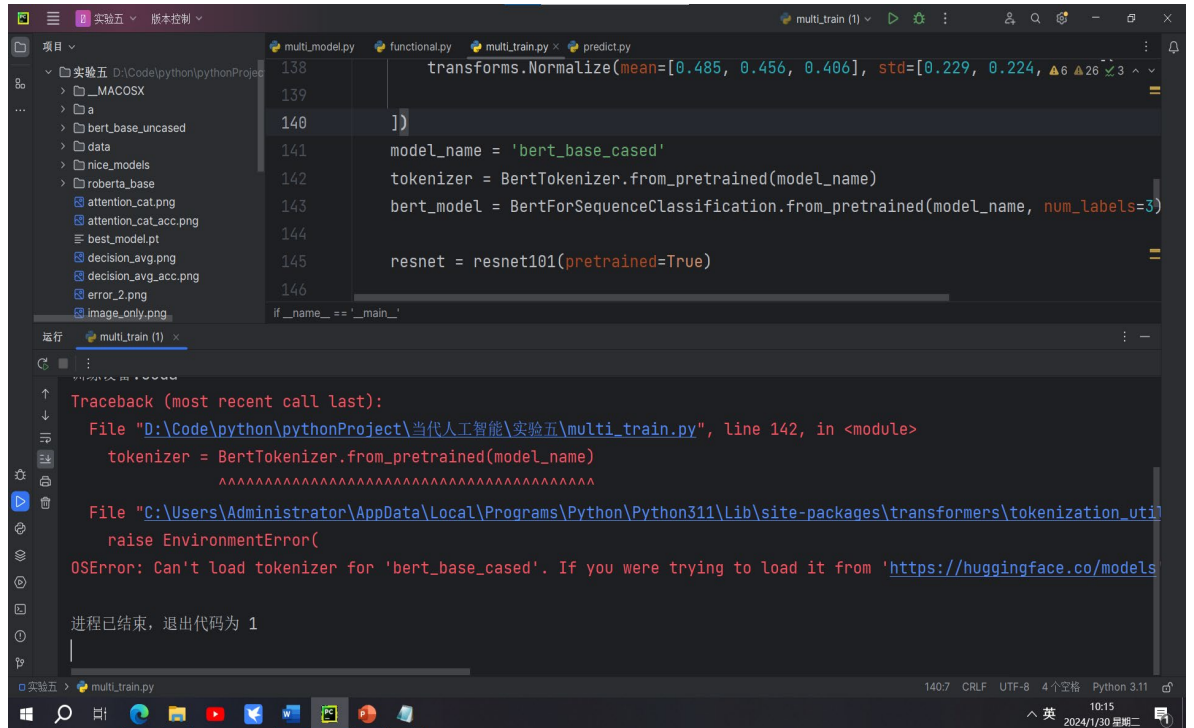
```

进程已结束, 退出代码为 0

多模态模型能充分利用多个模态的数据, 从中获取最全面的信息进行预测, 预测准确率比仅使用任意单一模态数据的效果都更高。

四、遇到的 bug

1. 无法载入与训练模型

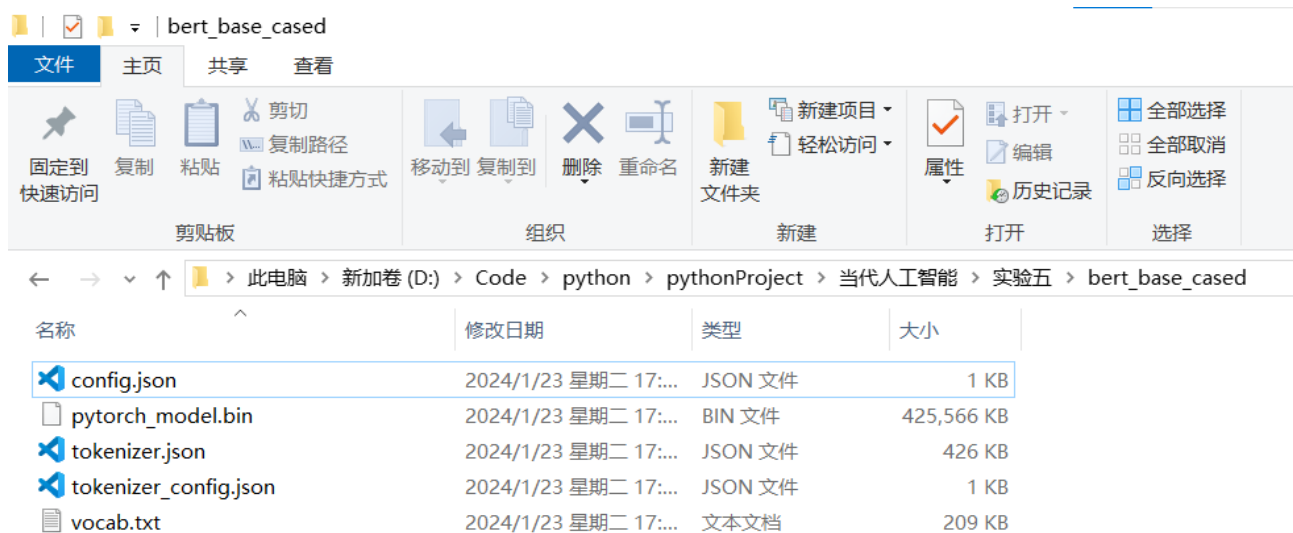


```
138 transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.226])
139
140 ]
141
142 model_name = 'bert_base_cased'
143 tokenizer = BertTokenizer.from_pretrained(model_name)
144 bert_model = BertForSequenceClassification.from_pretrained(model_name, num_labels=3)
145
146 resnet = resnet101(pretrained=True)
147
148 if __name__ == '__main__':
149     main()
150
```

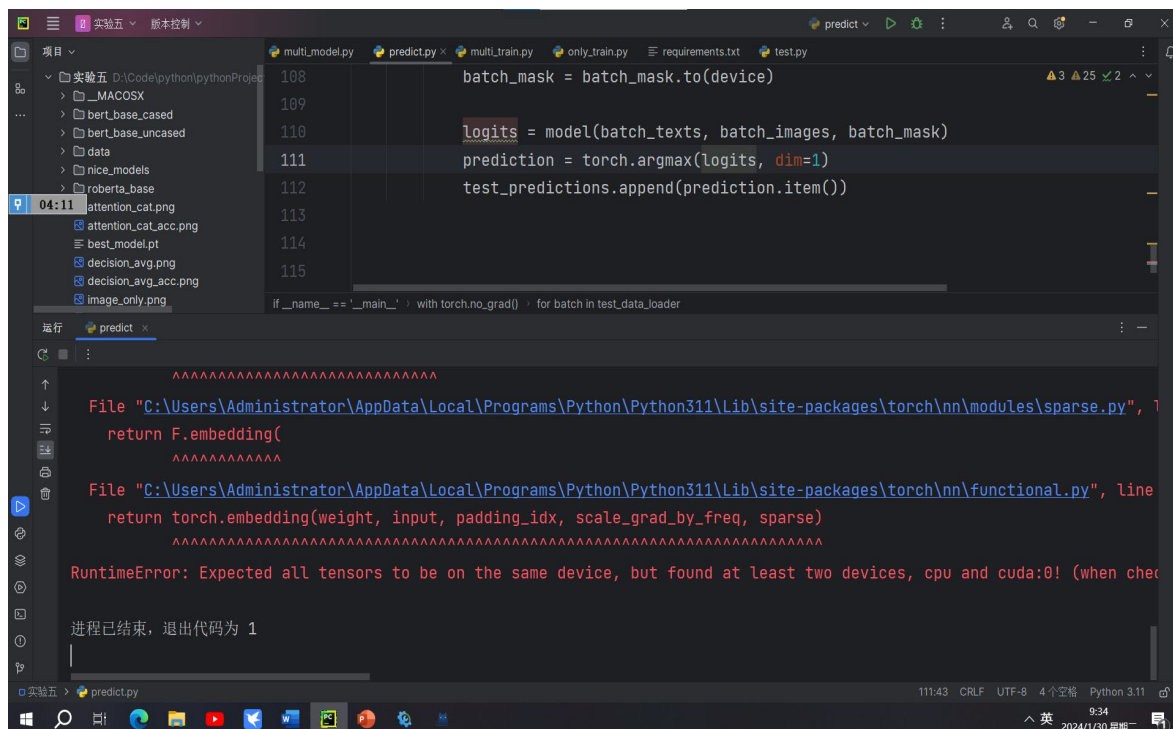
```
Traceback (most recent call last):
  File "D:\Code\python\pythonProject\当代人工智能\实验五\multi_train.py", line 142, in <module>
    tokenizer = BertTokenizer.from_pretrained(model_name)
    ~~~~~
  File "C:\Users\Administrator\AppData\Local\Programs\Python\Python311\Lib\site-packages\transformers\tokenization_utils.py", line 100, in from_pretrained
    raise EnvironmentError(
OSError: Can't load tokenizer for 'bert_base_cased'. If you were trying to load it from 'https://huggingface.co/models', please make sure you have the correct path to the tokenizer files.

进程已结束，退出代码为 1
```

由于网络问题无法直接访问 **huggingface** 下载预训练模型，于是我从其他渠道手动下载模型，模型文件夹位于项目根目录下。



2.模型载入错误导致参数未转移到目标设备上



检查了一遍后发现模型初始化后没有移动到对应设备上，导致前向计算时模型参数在 cpu 上而输入的数据在 gpu 上。

```
1. model.to(device)
2. model.load_state_dict(torch.load('best_model.pt',map_location=lambda storage
    , loc: storage.cuda(0)),)
```

将模型移动到 gpu 上并在载入模型参数时指定设备为 gpu 即可。