多模型融合情感分析·实验报告

汪柔柔

10215501439

目录

[实验环境 1](#_Toc157609448)

[实验目的 2](#_Toc157609449)

[实验思路 2](#_Toc157609450)

[实验过程 3](#_Toc157609451)

[实验结果 3](#_Toc157609452)

[思考总结 4](#_Toc157609453)

[设计亮点： 4](#_Toc157609454)

[执行流程 5](#_Toc157609455)

[参考库 5](#_Toc157609456)

[遇到的问题 5](#_Toc157609457)

[1、import paddle 5](#_Toc157609458)

[2. 向量维度出错 6](#_Toc157609459)

[问题三：多线程相关的索引越界问题 7](#_Toc157609460)

[问题四：验证集的划分： 10](#_Toc157609461)

[问题5：无法打开文件数量过多 12](#_Toc157609462)

[问题7：无法在字典中找到对应数字 16](#_Toc157609463)

[问题8：随着训练的增加，表现得越来越差 17](#_Toc157609464)

[问题9：requirements.txt文件生成（vscode环境下） 18](#_Toc157609465)

[错误10：索引错误 19](#_Toc157609466)

[问题11：最后一个批次问题 20](#_Toc157609467)

# 实验环境

Windows

jupyter

Python3.11

```

paddlepaddle>=2.0.0

numpy

tqdm

scikit-learn

torchvision

```

# 实验目的

本实验旨在探索多模态融合技术在情感分析任务中的应用。

具体来说，是基于给定的文本和图像配对数据集，构建一个多模态模型以预测对应的情感标签（例如，正面、中性或负面）。

通过该实验，我们期望验证文本与图像信息相结合是否能提高情感识别的准确性和鲁棒性。

# 实验思路

**1. 数据预处理：**首先对文本数据进行分词、编码，并将图像数据转化为合适的特征表示。

**2. 模型设计：**构建两个单独的模型来处理文本和图像输入。对于文本部分，使用卷积神经网络（CNN）结合嵌入层和全连接层；对于图像部分，采用ResNet或其他深度卷积神经网络提取特征。然后，将两个模态的输出特征进行融合，如使用注意力机制或简单拼接，最后通过一个全连接层进行分类。

**3. 训练与评估：**训练多模态融合模型并在验证集上评估性能，同时对比只使用文本或图像数据时的模型表现，即进行消融实验。

**4. 性能分析与优化：**针对实验结果进行深入分析，探讨模型的亮点以及可能存在的问题，并尝试进一步优化模型结构。

# 实验过程

1. 加载并预处理数据，包括文本序列化和图像特征提取。

2. 定义TextModel和ImageModel，分别用于处理文本和图像输入。（放在最后比较验证）

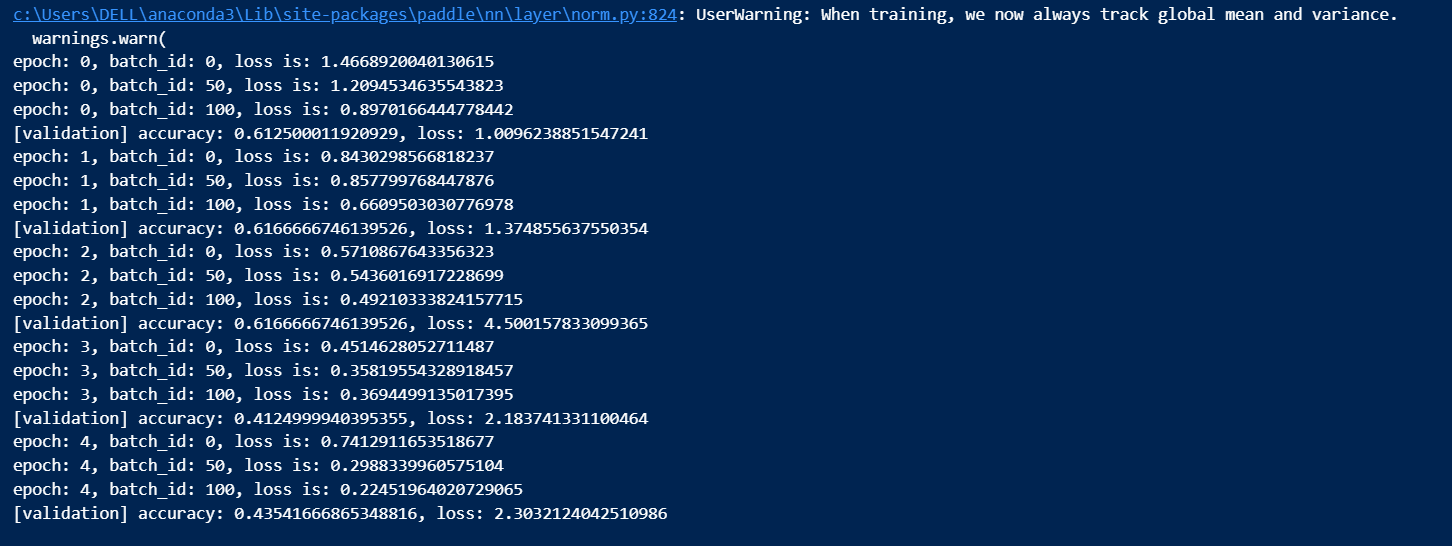
3. 设计一个多模态融合模块，将两种模态的信息整合在一起。

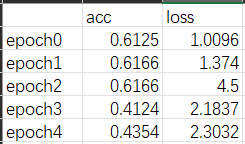
4. 编写训练脚本，实现模型的训练、验证和保存。

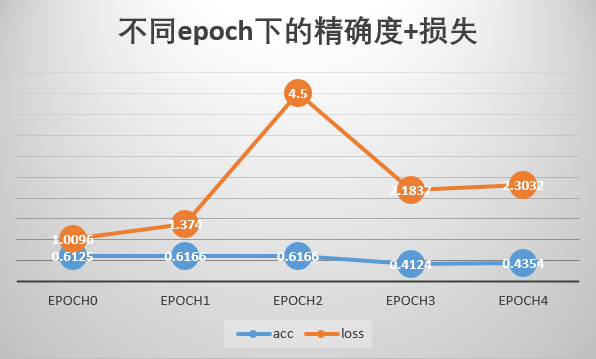
5. 执行消融实验，比较单模态和多模态模型在验证集上的性能差异。

# 实验结果（验证集上的）

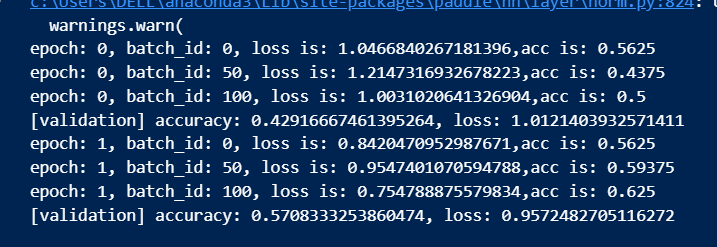
如下图所示，Epoch选择为2个最佳，因为损失最少，准确率相对最高，综合表现最好。



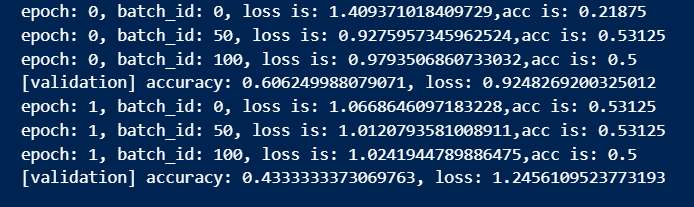




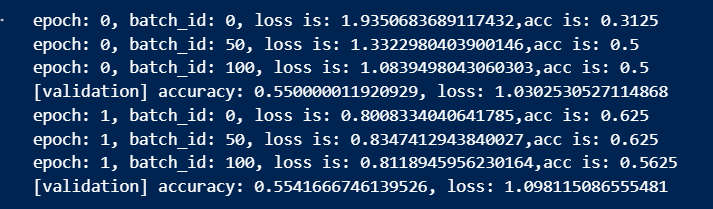
多模态融合模型结果（epoch=2）

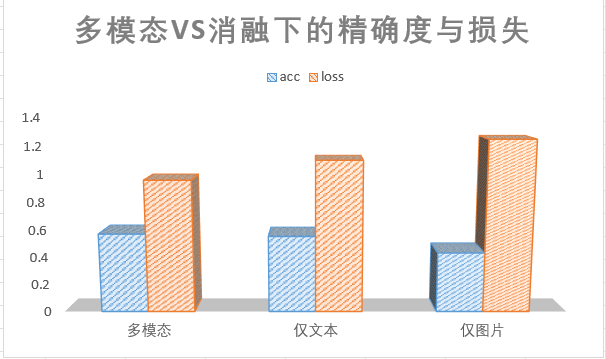
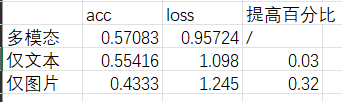


下图仅图片模型结果（epoch=2）



下图仅文本模型结果（epoch=2）





- 多模态融合模型在验证集上获得了显著优于仅依赖文本或图像模型的性能，准确率分别提高了约32%和3%。

- 文本模型和图像模型在验证集上的准确率分别为55.4%和43.3%，体现了不同模态各自的有效性。

# 思考总结

本次实验表明，多模态融合能够有效提升情感分析任务的表现，说明视觉和语言信息的结合可以更全面地捕捉到用户的情感状态。

# 设计亮点：

1. 模型设计中的亮点在于采用了独立且强大的文本和图像处理模块，并引入了一种有效的融合策略。未来可以考虑改进融合机制，比如利用更复杂的注意力机制，进一步挖掘文本与图像之间的关联。
2. 生成了例如test\_loader数据加载器，可以有效简洁的加载数据。
3. 对于test的label将无效的null标签更改为对应序号，对于后续的对比检查操作有所裨益。
4. 在划分训练集和验证集的时候，采用了1/8的抽取策略，可以避免由于原始数据分层带来的偏颇。
5. 在打开文本数据时，采用了多种常见的编码方式权衡效率，避免了由于utf-8单一编码方式带来的未知错误。
6. 在实验中多次设置try-exception，有效处理错误的出现。

# 执行流程

1. 安装所需依赖包：`pip install -r requirements.txt`

2. 运行`10215501439.jupyter`启动训练和评估、预测过程。

3、生成一系列文件，在dataset中生成updated\_test2.txt文件。（其余例如test2.txt只是辅助）(不过缺少第一行，我手动添加，最后提交的文件为test.txt为完整版)

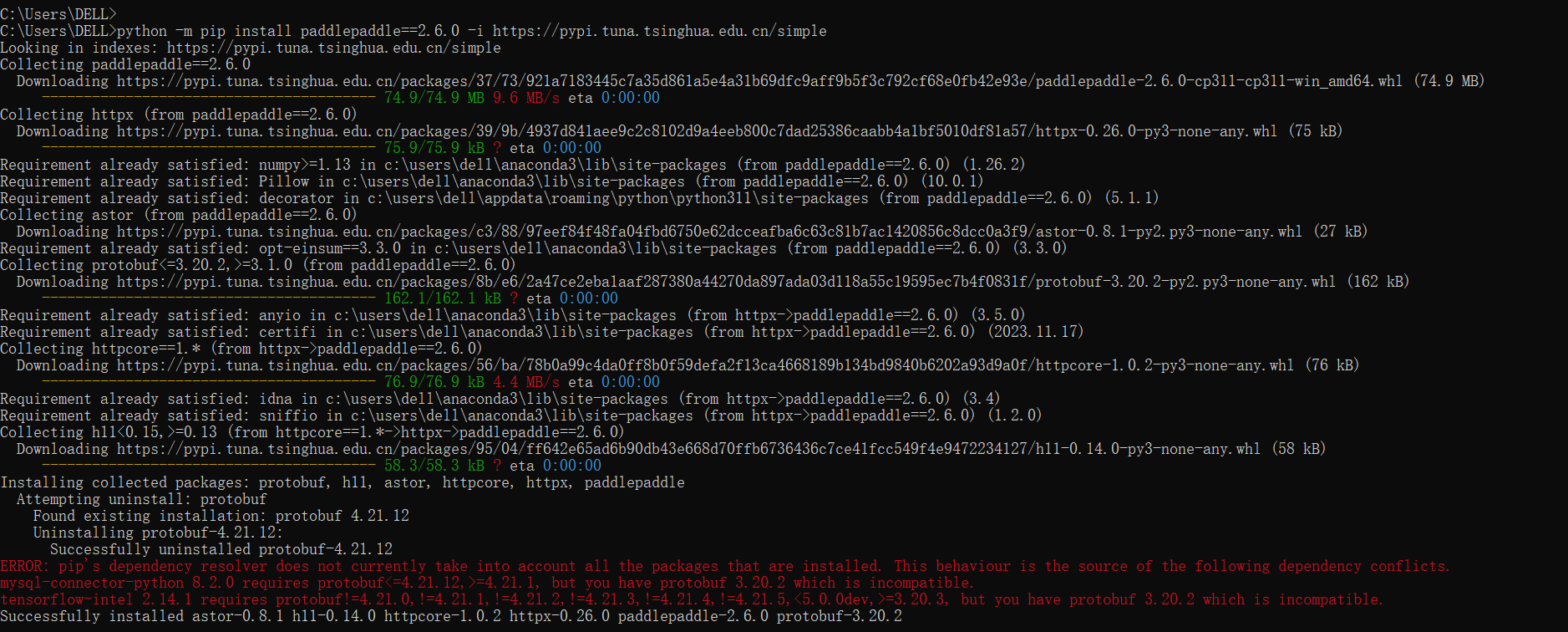
# 参考库

1. **深度学习框架**：
   * paddlepaddle：这是阿里云开源的深度学习框架，用于构建和训练模型。
   * 或者 PyTorch：由Facebook AI Research开发的流行深度学习框架，它提供了灵活且强大的动态计算图功能。
2. **数据处理与加载**：
   * paddle.io.DataLoader 或 torch.utils.data.DataLoader：这两个类分别对应PaddlePaddle和PyTorch的数据加载器，用于从数据集中高效地批量读取数据并传入模型。
3. **预处理与嵌入层**：
   * paddle.nn.Embedding 或 torch.nn.Embedding：用于将文本词汇映射为稠密向量表示。
   * 可能还会使用分词库如 jieba（对于中文）或 nltk（对于英文）进行文本预处理。
4. **图像处理**：
   * torchvision.transforms：如果在PyTorch环境下工作，这个库包含了一系列对图像数据进行预处理和转换的操作。
   * paddle.vision.transforms（假设存在，未验证）：PaddlePaddle环境中可能存在的类似图像处理工具包。
5. **模型构建**：
   * 构建CNN模型时会用到框架内的卷积层、全连接层等模块，例如 paddle.nn.Conv2D、paddle.nn.Linear 或 torch.nn.Conv2d 和 torch.nn.Linear。
6. **优化器**：
   * 代码中提到了 paddle.optimizer.Adam，这是Adam优化器在PaddlePaddle中的实现，PyTorch中对应的为 torch.optim.Adam。
7. **损失函数与评估指标**：
   * 使用了 paddle.nn.functional.cross\_entropy 计算交叉熵损失，在PyTorch中相应的是 torch.nn.functional.cross\_entropy。
   * 评估准确率时用到了 paddle.metric.accuracy，在PyTorch中可以通过比较预测值与真实标签来手动计算准确率。

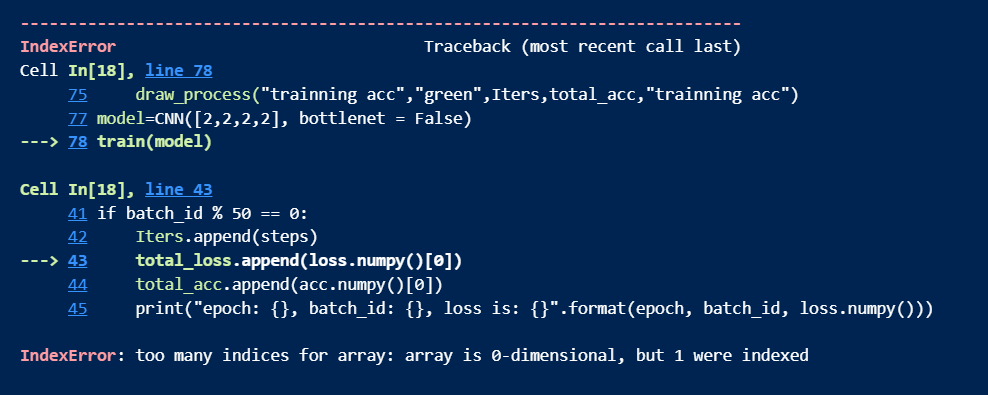
# 遇到的问题

## 1、import paddle

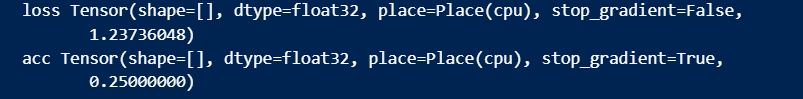
python -m pip install paddlepaddle==2.6.0 -i https://pypi.tuna.tsinghua.edu.cn/simple

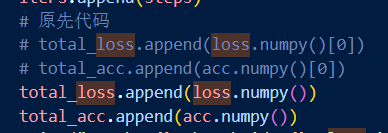


## 2. 向量维度出错

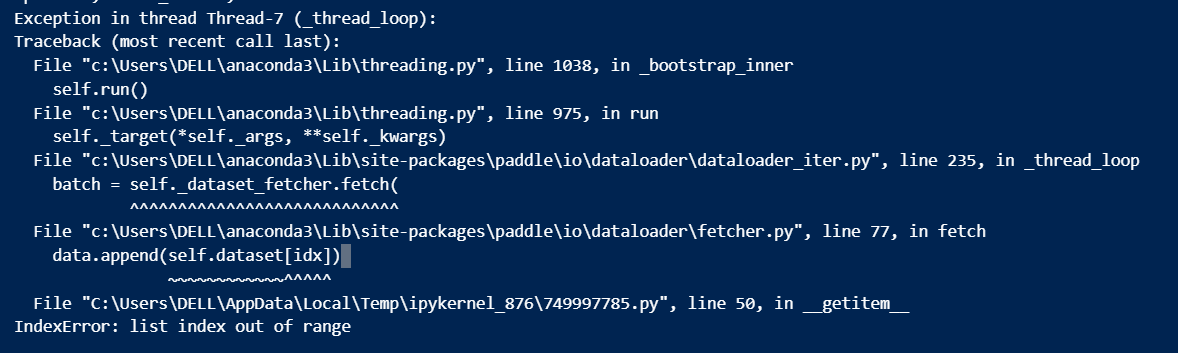


输出：





## 问题三：多线程相关的索引越界问题



尝试修改：

train\_loader = paddle.io.DataLoader(..., num\_workers=multiprocessing.cpu\_count() - 1)

test\_loader = paddle.io.DataLoader(..., num\_workers=multiprocessing.cpu\_count() - 1)





尝试改进：

为了防止`IndexError: list index out of range`异常，您需要在访问列表索引之前检查索引是否有效。以下是对`\_\_getitem\_\_`方法的改进版本：

```python

from PIL import Image

import transforms

class CustomDataset(Dataset):

def \_\_init\_\_(self, images, all\_data, transform=None):

self.images = images

self.all\_data = all\_data

self.transform = transform

def \_\_len\_\_(self):

return len(self.images)

def \_\_getitem\_\_(self, index):

if index < 0 or index >= len(self.images):

raise IndexError(f"Index {index} is out of range for the dataset.")

data, label = self.all\_data[index]

image\_path = self.images[index]

# Ensure the image exists and can be opened

try:

img = Image.open(image\_path)

except FileNotFoundError:

print(f"Warning: The file '{image\_path}' was not found. Returning a placeholder.")

img = Image.new('RGB', (224, 224), color=(0, 0, 0))

# Resize the image to 224x224

img = img.resize((224, 224))

# Apply any transformations, if defined

if self.transform is not None:

img = self.transform(img)

# Convert the image to a tensor

img = transforms.ToTensor()(img)

return img, data, label

```

这里我们做了如下改进：

1. 添加了对`index`值的范围检查。

2. 在尝试打开图像文件时使用`try-except`结构捕获可能发生的`FileNotFoundError`异常，并提供一个占位符图像。

3. 如果定义了数据增强或转换操作，确保在将图像转为张量前应用它们。

## 问题四：验证集的划分：

def create\_data\_list(data\_path, train\_path, val\_path, dict\_path):

    dict\_txt = load\_vocab(dict\_path)

    #print(dict\_txt)

    with open(data\_path, 'r', encoding='utf-8') as f\_data:

        lines = f\_data.readlines()

        print(len(lines))

        #print(lines)

    i = 0

    maxlen = 0

    with open(train\_path, 'a', encoding='utf-8') as f\_train, \

         open(val\_path, 'a', encoding='utf-8') as f\_val:

        for line in contents:

        # for line in lines:

            words = line.split('\t')[-1].replace('\n', '')

            #print(words)

            maxlen = max(maxlen, len(words))

            #label = line.split('\t')[0]

            label=str(labels[i])

            labs = f\_write\_txt(words, dict\_txt, label)

            # 每8个 抽取一个数据用于验证

            if i % 8 == 0:

                f\_val.write(labs)

            else:

                f\_train.write(labs)

            i += 1

    print("数据列表生成完成！")

    print(maxlen)

train\_path="train\_data.txt"

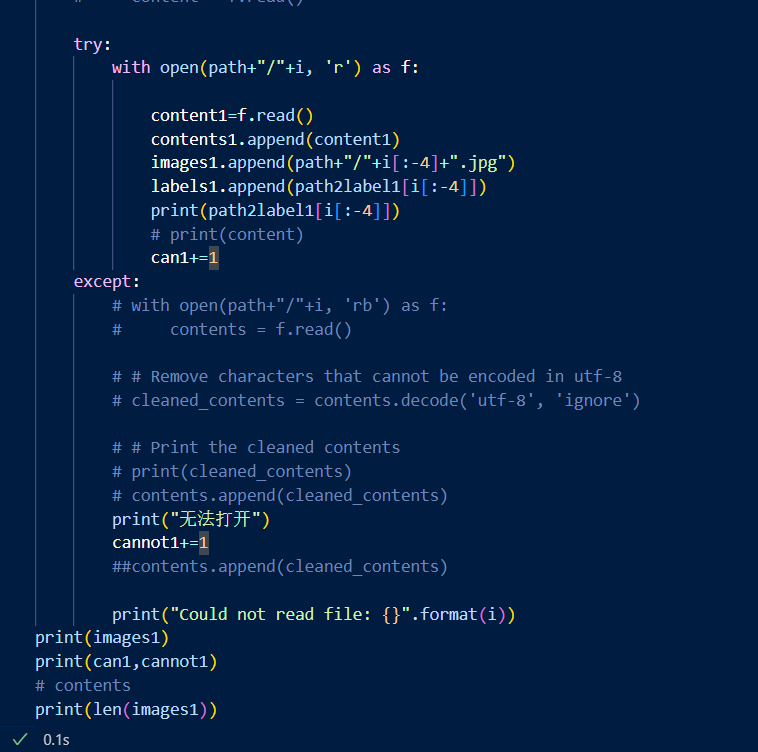
val\_path="val\_data.txt"  # 需要先定义验证集路径

create\_data\_list(all\_data\_path, train\_path, val\_path, dict\_path)

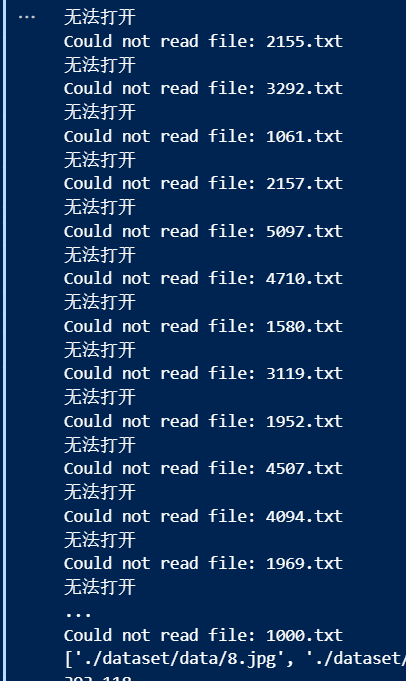
请确保自定义的`CustomDataset`类继承自`torch.utils.data.Dataset`。同时，请根据您的实际需求调整错误处理和占位符图像部分的代码。

## 问题5：无法打开文件数量过多

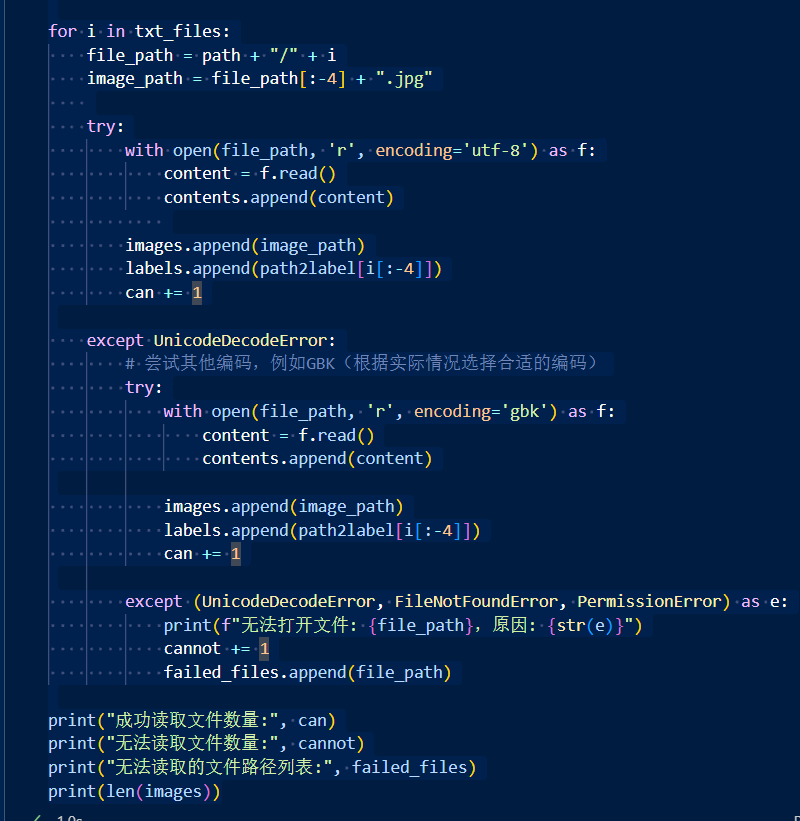
原代码：



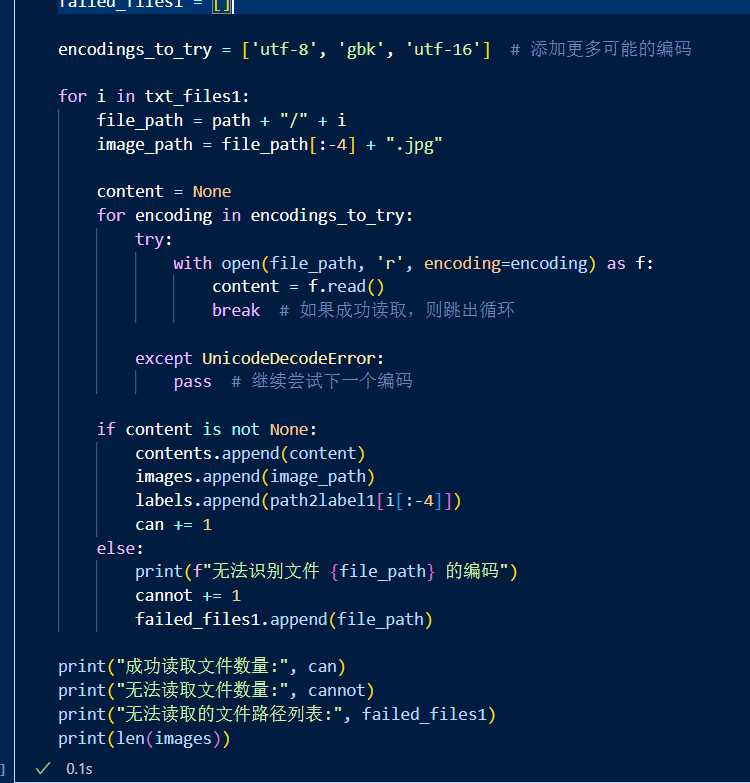
输出情况：

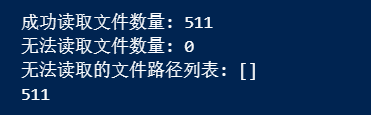


改进代码：

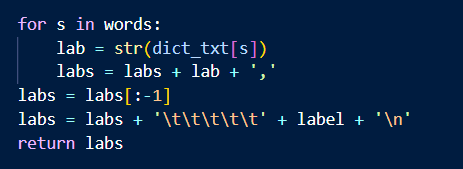


最终代码：

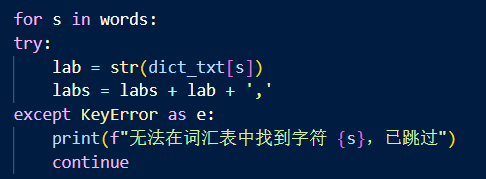


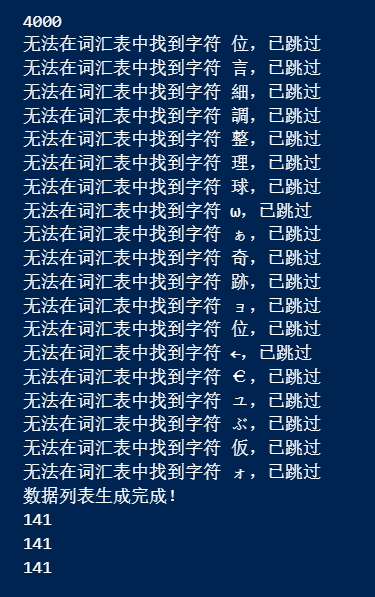


## 问题7：无法在字典中找到对应数字

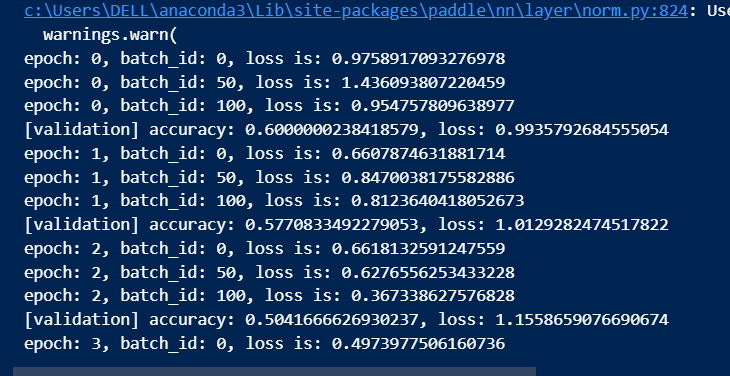


修改后



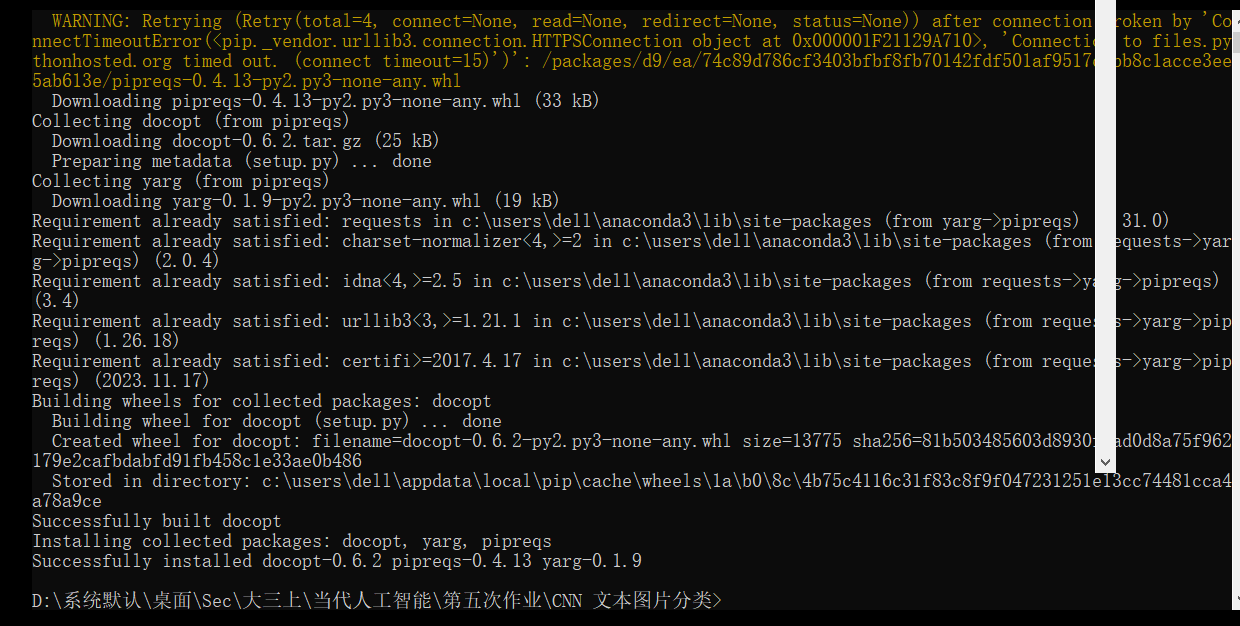


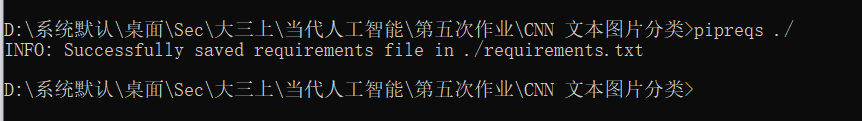
## 问题8：随着训练的增加，表现得越来越差



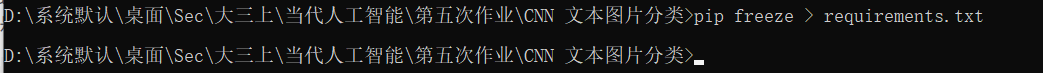
## 问题9：requirements.txt文件生成（vscode环境下）

pip install pipreqs

 D:\系统默认\桌面\Sec\大三上\当代人工智能\第五次作业\CNN 文本图片分类

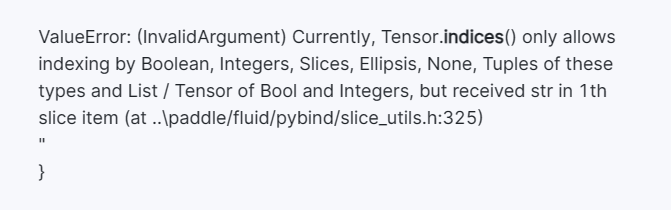


但是！使用pipreqs ./生成的requirements.txt文件是空白的。所以使用了另一个命令



[win10下使用pipreqs提示不是内部或外部命令 - SunboyL - 博客园 (cnblogs.com)](https://www.cnblogs.com/SunboyL/p/14951295.html)

## 错误10：索引错误



这里的问题在于data[0]可能不是一个字典结构，而是张量或其他不支持通过字符串索引的对象。

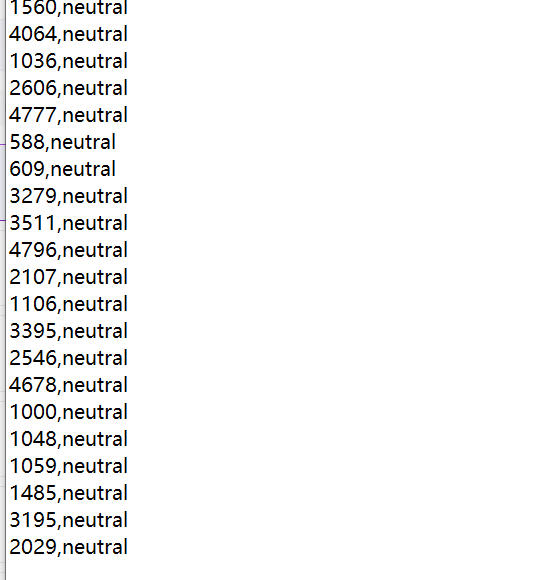
原代码

image\_path, input\_sent, true\_label = data[0]['image\_path'], data[1], data[2]

现代码

image\_path, input\_sent, true\_label = data[0], data[1], data[2]

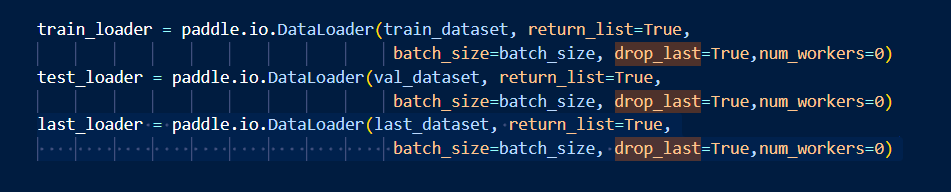
## 问题11：最后一个批次问题



原因：drop\_last设置为了true。

更改：

在创建DataLoader时设置drop\_last=False，这样即使最后一个批次不满也会被包含进来。



但是不能更改drop选项，否则会报如下错误。

ValueError: (InvalidArgument) Input(Y) has error dim. Y'dims[0] must be equal to 96, but received Y'dims[0] is 4736.

[Hint: Expected y\_dims[y\_ndim - 2] == K, but received y\_dims[y\_ndim - 2]:4736 != K:96.] (at ..\paddle/phi/kernels/impl/matmul\_kernel\_impl.h:329)

[operator < linear > error]