实验五 多模态情感分析

一、实验任务

三分类任务：给定配对的文本和图像，设计一个多模态融合模型，预测测试集数据对应的情感标签（`positive`，`neutral`，`negative`），最后使用消融实验，检验模型对文字和图片的训练情况。

github

二、实验数据处理

`data_processing.py`

基础内容

1. 初始化

涵盖了我们在整个模型训练中需要的内容：

- 初始化BERT模型所需的的tokenizer和最大序列长度max_seq_length。
- 定义标签映射 `tag_map` ('negative': 0, 'neutral': 1, 'positive': 2)，便于模型的训练和预测。
- 提取传入文件的 `guid` 和 `tag` 列，并检查 `guid` 是否有效，生成图像文件路径列表。

2. 获取单个样本 `__getitem__`

该功能需要根据传入的索引 `idx`，提取出 `guid` 和 `tag`。

- 判断 `tag` 是否有效，将缺失/无效值替换成negtive，在对应 `tag_map` 直接转换成0、1、2。在**训练集**中，标签都是有效的；但在**测试集**中，原本的值为null，直接替换成 `negative` 不会有预测上的影响。
- 分别加载每一个 `guid` 对应的图像和文本内容。
- 对文本进行BERT的 `tokenizer` 处理，进行编码，使模型能够接受并训练。
- 返回**字典**，包括图像、文本生成的token的id、掩码attention_mask以及tags，注意tags转换成long的形式进行**数据类型统一**。

3. 加载文本和图像 `_load_text` & `_load_image`

加载 `guid` 对应的文本和图片：**文本**使用utf-8解码，对应 `guid` 和文本；**图片**定位图像文件，对应 `guid` 和图像。这一部分在训练后，经过尝试，我们准备在这里采用**数据增强**的优化方式。

亮点优化：数据增强

*为什么选择**设计**数据增强？*

在没有数据增强的情况下，我们发现经过若干epoch后，训练集的acc能够**从0.6上升到0.99**，而验证集只能**从0.6上升到0.7左右**，甚至在第6个epoch到达峰值后就呈现出轻微的下降趋势（在0.01范围内）。

因此，想到在**训练模式**通过让模型学习**不同的视角、光照、旋转的图像变换**以及**经过同义词替换的文本**，来缓解模型的*过拟合*情况并提高模型的*泛化能力*。

如何进行数据增强?

图像：裁剪（模拟不同缩放比例）；翻转、旋转图像；调整图像亮度、对比度饱和度等（模拟不同光照）。

文本： `synonym_replacement` 函数替换同义词，避免模型过于依赖特定的词汇或表达方式。

*相关结果在**实验结果**中进行分析*

三、模型设计

`model.py`
采用经典的预训练模型bert和resnet50，通过多头注意力机制和Transformer编码器融合文本和图像特征。

文本部分： `BERTEmotionModel`
采用 `HuggingFace` 的 `transformer` 加载预训练模型 `bert-base-uncased` 提取文本特征，将BERT输出特征 `hidden_size = 768` 映射到更小256维度中，采取dropout防止过拟合，最后使用激活函数。**输出每个token的隐藏状态 `[batch_size, seq_len, hidden_size]` 以及全连接层后的输出 `[batch_size, hidden_size]`**

图像部分： `ResNetEmotionModel`
使用 `torchvision.models` 加载预训练的ResNet-50模型，提取图像的特征。
输入： `[batch_size, 3, 224, 224]` 图像内容。
输出： `[batch_size, 2048]` 特征向量。

一开始我们想要选择更高层次的ResNet-101，即使用更大的残差块，但训练效率更低，正确率提升不明显，也有过拟合的情况存在，因此最后还是选择回ResNet-50。

多模态融合

核心组件：

- **位置编码 `position_embeddings`** ： 为文本特征添加位置编码以保留序列信息，维度 `[max_seq_length, hidden_size=256]` 。
- **特征对齐 `embedding`** ： 使用全连接层将图像特征映射到与文本特征相同的维度（256）。

- **多头注意力机制** `MultiheadAttention`： 对文本和图像特征进行注意力计算，捕捉模态间的交互信息，采用0.1的dropout率。
- **Transformer编码器** `TransformerEncoder`： 使用多层 Transformer 编码器进一步融合特征，每层包括**自注意力机制**和**前馈神经网络**。
- **分类层** `fc`： 最后映射到三类情感标签。

在该部分中，面对仅图像、仅文本和文本图像结合不同情况的多模态处理如下：

- **仅图像**：输入图像数据后，使用resnet进行特征提取，得到特征后将维度从 `[batch_size,256]` 调整为 `[1, batch_size, 256]` ，以便后续与文本特征拼接。
- **仅文本**：将文本数据与attention_mask一同输入bert模型提取文本特征，得到text_features后将维度从 `[batch_size,256]` 调整为 `[1, batch_size, 256]` ，以便后续与图像特征拼接。
- **多模态融合**：若两者结合，则使用torch生成位置id `position_ids` ，随即生成**位置编码** `position_embeddings` ，将位置编码加到文本特征上；获得新的文本特征后，将文本和图像特征在**序列维度**进行拼接，接着即可使用**多头注意力机制**与transformer编码器***融合特征***，最终得到均值 `final_feature` .

四、实验结果

`test.py` （消融） & `evaluate.py` （融合特征）

多模态融合模型验证集结果与消融实验结果

1. 原始结果

未使用数据增强，训练集：验证集为8：2的情况下，*得到的准确率为：*

```
image_only - Train Loss: 0.0339, Train Accuracy: 0.9944
image_only - Validation Loss: 1.2125, Validation Accuracy: 0.6212
Training text_only model...
text_only - Train Loss: 0.1501, Train Accuracy: 0.9537
text_only - Validation Loss: 1.2717, Validation Accuracy: 0.6238
Training multimodal model...
multimodal - Train Loss: 0.0312, Train Accuracy: 0.9909
multimodal - Validation Loss: 1.5288, Validation Accuracy: 0.7013
Training complete.
Best image_only model path: C:\Users\admin\Desktop\project\result\best_image_only_model_20250120_131737.pth
Best text_only model path: C:\Users\admin\Desktop\project\result\best_text_only_model_20250120_131943.pth
Best multimodal model path: C:\Users\admin\Desktop\project\result\best_multimodal_model_20250120_134341.pth
```

在该情况下，可以看到无论是消融还是多模态融合，训练时都能够达到一个很高的水平，其中仅图像的情况和多融合模型的**训练**准确度达到了0.99，仅文本情况达到0.95。

在训练过程中，我们发现训练准确度持续上升且情况优异，而验证准确度保持在0.7上下轻微浮动，猜想在这种训练集验证集划分的情况下，模型会产生过拟合的情况，**泛化能力不好**，因此我们选择下调比例，进行训练。

2. 最佳结果 训练集：验证集为7：3

预测文件： `./result/predictions_best.txt`
调整训练集-验证集比例后的结果：

```
image_only - Train Loss: 0.7880, Train Accuracy: 0.6293
image_only - Validation Loss: 0.7879, Validation Accuracy: 0.6492
text_only - Train Loss: 0.6246, Train Accuracy: 0.7786
text_only - Validation Loss: 0.7264, Validation Accuracy: 0.7133
multimodal - Train Loss: 0.5415, Train Accuracy: 0.7893
multimodal - Validation Loss: 0.6647, Validation Accuracy: 0.7325
```

最高值如上，整体大约能够提高4-5个百分点。其中，消融实验中，仅文本情况的提升情况最好。***获得最佳的多模态验证集准确率：73.25%***

比例为8:2时，模型会产生过拟合情况，加之我们拥有的*文本数据集*的信息含量都较少，更加容易受到过拟合的影响，因此在下调比例后，获得了很大的验证提升。

合理性：分别对 `train.txt` 和 `test_without_label.txt` 的标签进行统计，发现最终的分类结果与训练集中的标签分布一致。

Dataset	pos	neg	neu
train	2388	1193	419
test	318	176	17
ratio	6	3	1

3. 数据增强后的结果 `./result/predictions_with_ablation.txt`

为了进一步消除过拟合给模型带来的影响，采用数据增强的方式进行处理。*使用后，最佳的验证集准确率如下：*

```
multimodal - Train Loss: 0.4911, Train Accuracy: 0.8165
multimodal - Validation Loss: 0.7765, Validation Accuracy: 0.7264
```

在模型学习的过程中，**没有数据增强**处理时，过拟合在**epoch 4**开始就已经出现，可以参考上述实验结果，出现最佳验证集准确率时，训练集准确率仅为**0.78**；而使用了**数据增强**处理时，验证集准确率虽然比*不处理*的情况低，但对比训练集准确率和所有epoch的训练过程可以发现，该方法**有效缓解了使用单一数据集进行训练时出现的过拟合，同时提高了模型的稳定性。**

备注：
*所有epoch的训练过程截图较难，未展示。当**训练集：验证集为7:3**时大致的训练状况如下：*

***原始处理**：验证集loss持续上升，在10个epoch内到达了1.5左右，且验证集准确率达到0.73后开始持续下降，至第10个epoch时仅有0.71左右*
***数据增强**：验证集loss维持在0.8以内，且验证集准确率始终保持在0.72-0.73的区间中*

五、遇到的问题

1. 预处理问题 避免特殊字符

`guid = str(guid).split(',')[0]` 在这一步中要注意，直接取逗号前的部分，避免特殊字符影响文件路径，如果进行更进一步的判断则可能会出现将数据中一整行都纳入文件名的可能，
如 `123,negtive.txt` 。（同理在 `def _load_image(self, guid):` 中也会有类似的问题，遇到时需要注意）

```
guid = str(guid).split(',')[0]
text_file_path = os.path.join(self.image_dir, f"{guid}.txt")
```

读数据，跳过第一行(`header=0`)，否则会将第一行的内容纳入文件名出现 `guid.txt` 。

```
self.data = pd.read_csv(text_file, delimiter=",", header=0, names=["guid", "tag"])
```

2. 预处理问题 NaN的处理

在进行 `__getitem__` 时，特别注意获取tag的处理，出现无效的tag（三种情感之外）时，直接将其设置为negtive（0），便于之后取用，否则将会出现NaN报错。

```
File "C:\Users\admin\Desktop\project\data_processing.py", line 48, in __getitem__
    assert 0 <= tag < 3, f"Invalid tag: {tag} at index {idx}"
AssertionError: Invalid tag: nan at index 134
```

3. 数据读取错误

text数据处理时偶尔会跳出编码错误的提示，但是由于只是个别且没有规律，我们直接采用 `errors="ignore"` 解决，不影响训练与预测。

六、总结

本次实验采用了本学期中所学的印象深刻的三个模型，分别是**文本模型bert**、**图像模型ResNet**以及**语义生成模型Transformer**，虽然现在的技术已经有了更加优秀有效的模型，但我认为调用和实现这些里程碑模型也是人工智能学习中比较重要的一步。

本次实验的数据集大小大概有5000，在深度学习中该数据集规模可能偏小，因此我们一开始就选择了较为简单的模型进行训练（同时，ResNet残差网络能够缓解深度学习中梯度消失或爆炸的情况），防止严重的过拟合。在训练结束后，我们通过观察结果并总结其不合理之处，及时优化了数据集的分配，并对数据集进行了多样化处理，保证简单模型也能够有较好的泛化能力。