基于 CLIP 的多模态情感分类

1 模型

本次实验的核心任务是把图片和文本编码成特征向量, 然后以某种方式融合, 最后用 MLP 进行分类. 因此, 我选择了 CLIP 模型完成本次实验, 该预训练模型基于文本-图片对的对比学习方法, 可以有效地完成本次实验的任务.

CLIP 模型在文本端基于 Transformer 架构编码, 图片端则是提供了 ResNet 和 Vision Transformer 可选. 由于该模型具有很强的 Zero-Shot 分类能力, 我们可以直接冻结其全部参数, 只训练接在后面的 MLP 分类层. 这样训练对显存的要求比较低, 因此在视觉侧我直接选用了最大的模型 ViT-L/14@336px, 这里-L 代表大尺寸版本, 有 24 层 Transformer,14 代表会把图片分为 14 像素的 patch 处理,336px 表示对输入图片的尺寸调整.

该模型会把图片和文本都编码为 768 维度的向量, 由于显存限制, 我并没有去微调 CLIP 的最后一层 Transformer, 所以我没有选择简单的接一个 768->3 的线性层直接进行分类, 而是加入了一个 256 维的隐层引入特征的非线性交互, 并进行 LayerNorm 和 GeLU 激活 (Transformer 系列模型常用的归一化方法和激活函数) 以及 Dropout.

对于特征融合方法, 我设置了三种方案:

- 直接拼接: 这是最符合直觉的, 因为文本和图片在我们看来是不直接相关的, 直接拼接出一个 768*2 维度的向量作为特征是最简单的想法, 类似于「后期融合」.
- 可学习的加权求和: 也是比较朴素的想法, 通过可学习参数调整图片和文本的权重, 从均为 0.5 开始, 使用 softmax 确保权重和为 1.
- 注意力拼接: 将图像和文本特征拼接成序列, 然后通过多头自注意力计算特征间的交互, 最后取平均得到融合特征.

对于注意力拼接, 除了可以使用和另外两种一样的分类器, 我们还可以尝试直接接一个 768->3 的线性层. 因为此时通过注意力融合已经引入了非线性, 并不一定需要 MLP 再引入非线性.

综上, 我们的模型架构如下:

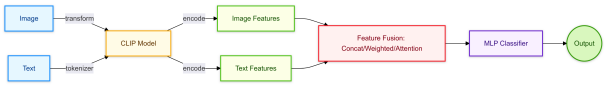

Figure 1 Model Architecture

其设计主要有以下亮点:

- 基于 CLIP 模型, 文本和图片统一性较高, 代码实现简洁, 并且可以获得较好的多模态性能, 同时维持可解释性
- 提供多种拼接策略可供对比研究, 且可以方便的进行消融实验
- 基于预训练模型引入一个 MLP 层, 保持训练过程高效的同时通过简单的非线性的引入实现较高的准确率并减少过拟合

2 训练

训练过程我主要使用了如下配置:

- AdamW 优化器, 默认的 0.01 权值衰减
- OneCycle 学习率调度器, 峰值为 1e-4(我们只是训练最后的 MLP, 不需要太大的学习率),0.3 比例预热, 余弦退火; 最多 10 个 epoch, 连续 2 个 epoch 验证集性能没有提升时早停
- MLP 中的 Dropout 我设置了较大的 0.5, 因为在实验中观察到了严重的训练集过拟合
- 64 的批大小, 需要约 4GB 显存
- 通过随机种子设置和 cudnn 配置确保实验结果的可复现性

*

*GitHub: Repository Address

- 对图片数据, 在 CLIP 自带预处理 (图片大小调整) 的基础上, 在训练时增加旋转, 水平翻转, 色彩调整和仿射变换等数据增强

观察训练过程, 第一个线性层的数据分布符合正态, 说明我们的参数初始化等配置正确.

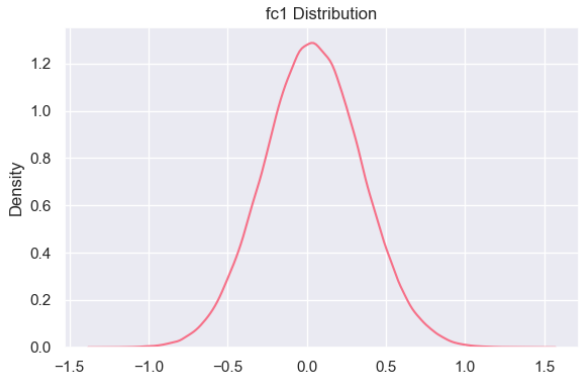

Figure 2 FC1 Distribution

本次实验的一个较大的问题是数据集严重不平衡, 具体分布如下:

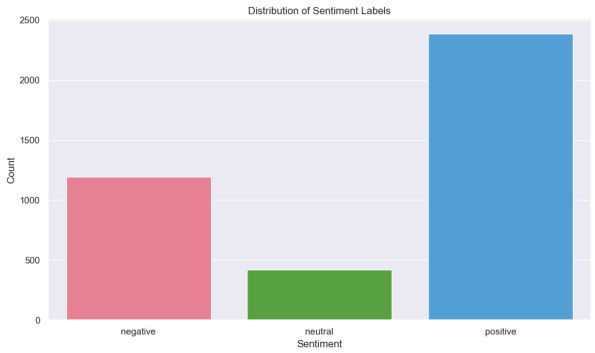

Figure 3 Dataset Label Distribution

可以看到最多的 positive 已经是最少的 neutral 类五倍了, 这必然导致模型很难学好 neutral 类的特征, 也给我们评估模型造成了困难. 我的假设是最终问题的比例分布类似于所给训练集, 并且我们最看重的是总体的 accuracy, 因此我的验证集是分层抽样而得的, 保持三个标签类别比例不变. 训练集有 4000 条, 测试集约 500 条, 因此验证集我也划分了 500 条, 即约 8:1:1. 训练集比例相对较高, 但由于我们总数据量有限, 继续缩减训练集可能会成为模型泛化能力的瓶颈. 训练过程中对该不平衡问题的矫正主要有两种思路:

1. 生成训练集时通过采样给予 neutral 类更大权重, 在一个 batch 内保持三类样本数量均衡
2. 在计算 loss 时给稀少类样本更大的权重

但基于我们对现实的假设和目标, 过度矫正这个问题会让模型不自然的预测太多的 neutral 类, 导致整体的 accuracy 下降. 因此我给的默认方案是方法二, 并且只给了 neutral 类很小的权重增益, 经实验, 这样做可以提高 neutral 类的 recall, 同时不会让总体的 accuracy 下降太多. 不过这是一个治标不治本的方法, 模型仍然会在 neutral 类上表现很差, 归根结底是总共就只有少量的 neutral 类样本, 模型的泛化能力上限受到制约. 此外, 我在计算交叉熵损失时应用了标签平滑来抑制模型过拟合.

3 实验

以直接拼接为例, 训练过程中训练损失和验证损失如下:

Figure 4 Loss

可以看到训练过程中损失下降基本比较合理. 由于我们给 MLP 设置了一个较大的 Dropout, 因此这里的训练损失会比较大, 但这也有效的抑制了模型的过拟合, 可以看到一直到第八个 epoch 验证集损失基本持续下降, 但后期大致不变, 而训练集损失还有持续下降的势头. 如果继续训练, 训练集上的准确率可以更高, 但验证集的就不一定了.

我们注意到这里模型在第 8 个 epoch 就早停了, 但此时验证集损失并没有出现明显的反增, 这是因为我采用了验证集上的加权 F1 作为早停的监控指标. 在后期交叉熵损失仍可以进一步下降, 但很多是因为模型更确定了他本来就正确分类的答案, 这可以降低交叉熵损失, 但对于总体的准确度并没有明显增益.

考虑到先前所说的类别不平衡问题, 我采用了加权平均的方式计算 precision,recall 和 F1, 而在这种情况下, 加权的 recall 和总体的 accuracy 是一致的;F1 综合考虑了 precision 和 recall, 因此用它作为指标最为全面. 验证集上各指标数据如下:

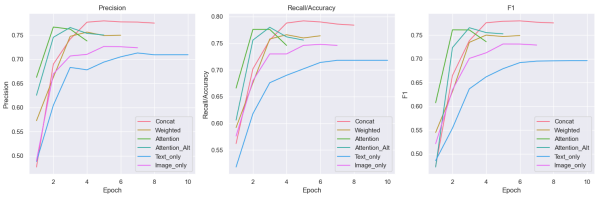

Figure 5 Metrics

这里 Attention_Alt 指的是直接对注意力融合后的特征用一个线性层分类的版本, 观察它和 Attention, 即再接一个带隐层的 MLP 的, 我们发现后者引入了更多非线性, 让模型可以更快收敛 (在第二个 epoch 就达到了峰值, 是所有模型里最快的), 但指标略差一些, 这提醒我们不要盲目引入复杂度. 少了这一层 MLP 后, 模型在一开始 (第一个 epoch) 的性能会比较差, 无法立刻拟合好特征, 但在后面可以调整过来. 通过引入注意力机制这样强大的非线性, 他们的收敛速度比其他方法都快.

在三种融合策略中, 我们可以看到大体上是拼接 > 注意力融合 > 加权求和 (F1:0.7799,0.7655,0.7500), 最好的拼接在验证集上可以取得约 79% 的 accuracy, 但他的收敛是最慢的 (特征维数是其他的两倍). 可见, 对于图片和文本特征向量, 如果我们把握不清楚他们之间到底是怎么交互的, 最简单有效的策略就是把他们分开来处理; 如果我们真的想要融合这两者的话, 基于注意力机制的融合比简单的加权求和会更准确.

对于消融实验, 只用图片或者只用文本都明显不如任何一种拼接方式,F1 分别只有 0.7312 和 0.6962. 因为 CLIP 模型基于文本和图片的对比学习方法, 而不是专门针对文本或图片的模型, 其消融实验结果比多模态融合结果更差是很自然的. 其中文本的结果相对较差, 并且更难收敛, 除去数据集本身的原因之外 (例如数据集中文本数据质量较低), 我认为 CLIP 在图片方面是更加侧重的.(它提供了多种视觉模型可选, 但在文本方面没有提供任何选择; 其文本能力应该不如 BERT 系列这样专门针对文本的预训练模型)

4 分析

4.1 错误案例分析

以拼接融合为例, 验证集上的错误案例有:

Figure 6 Error Cases

在这里我们同样可以意识到为什么消融实验的结果明显较差, 光看这些图片的话, 我们人类也很难对他进行情感分类. 矛盾的地方在于, 至少对我们人来说, 我们从文本出发应该是可以做出更好的分类, 但 CLIP 模型的图片能力更强, 文本方面则有所欠缺. 我们可以看到在这些错误案例上, 模型给出的置信度也基本不太高. 如果我们要设计一个非常高性能的模型, 我认为一方面模型的文本能力必须足够强, 这样才可以更好的近似人类的判断行为, 并且必须有一种足够高级的策略去融合图片和文本的特征, 这方面有待后续探索.

4.2 权重与注意力

我们设置了多种融合策略, 其中可学习的加权融合方法权重变化如下:

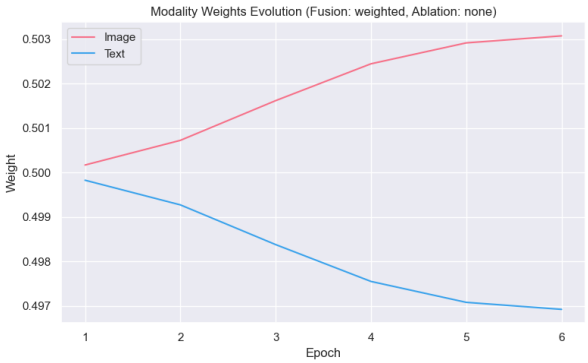

Figure 7 Weights Evolution

我们发现在训练过程中模型会更加强调图片特征. 在刚才的消融实验结果中我们也看到只用图片的优于只用文本的, 说明在本任务中, 虽然我们不能直接说图像特征比文本特征更有用, 但用合适的模型提取图片特征进行准确分类相比于文本特

征是更容易实现的.

类似的, 对于注意力融合, 文本和图像的注意力分数如下:

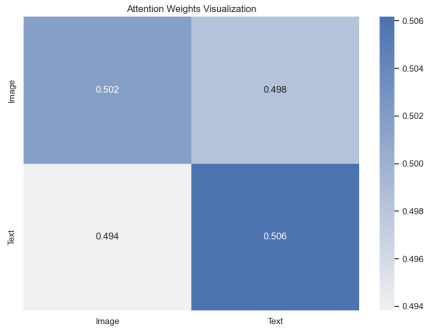

Figure 8 Attention Weights

我们发现模型基本上认为文本和图片特征同等重要, 没有明显的偏好. 刚才的加权求和中我们也可以看到虽然图片的比例在增加, 但变化仍然是轻微的. 说明总体上我们的模型并没有对这两个特征进行较大的区分, 如何做到这一点, 即实现这两种特征真正高度有意义的融合, 也是我们未来可以进一步研究的.

4.3 混淆矩阵与置信度

以拼接融合为例, 验证集上混淆矩阵如下:

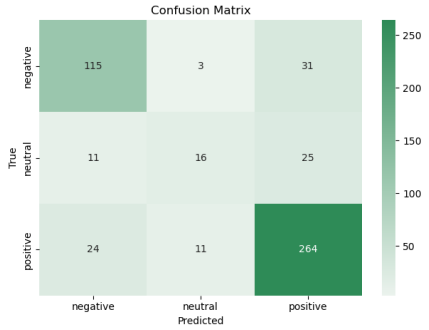

Figure 9 Confusion Matrix

可以看到模型的主要问题是对 neutral 类的预测比较不理想, 该类的 recall 只有约 0.3077(作为对比, 另外两类分别是 0.7718 和 0.8829). 这个数据还是基于我们对损失计算中类别加权之后的了, 否则该类的 recall 可能不到 0.2. 并且 precision 也比较低, 只有 0.5333. 如果我们继续加大该类的权重, 虽然 recall 可以提高, 但 precision 会继续降低, 即模型预测了很多错误的 neutral, 而这种情况是我们更不愿意看到的. 妥协之后只能是略微增加 neutral 类的权重, 尽可能不显著损失总体的 accuracy. 归根结底, 数据集 neutral 类的样本本身就太少 (当然这符合现实世界的情况, 人们需要在社交媒体上分享的自然大多数是有明显情绪倾向的内容), 模型很难学好这一类的特征.

从另一个角度, 我们可以给出模型对各标签类别预测的置信度分布, 其核密度估计如下:

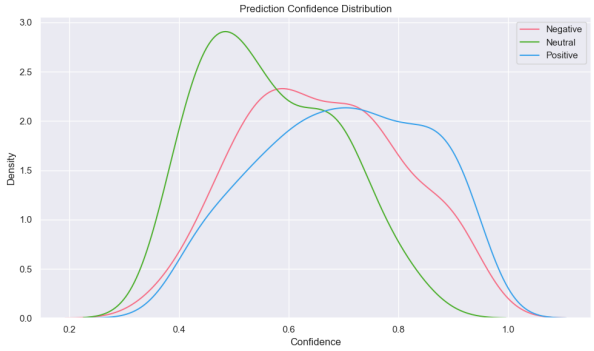

Figure 10 Confidence KDE

可以看到总体上模型对 neutral 类的置信度是比较低的, 集中在 0.6 以下并且明显右偏 (统计学意义上的 skewed), 随后依次是 negative 和 positive(这也是一开始的训练集样本分布的顺序). 最好的 positive 则左偏并且众数约有 0.7 至 0.8. 可见模型性能的根基仍然是训练数据集, 这一基本事实不会改变.

下图展示了模型在一个验证集 batch 上各类别的预测置信度情况:

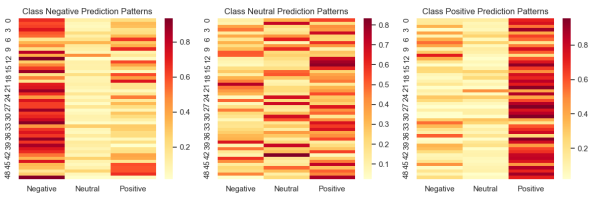

Figure 11 Prediction Pattern

可以看到对于 negative 和 positive, 大多是在正确类别有高置信度, 但 neutral 类就比较混乱.

如果我们把这个模型投入现实世界对社交媒体上的内容进行分类, 我们真的想要精准鉴别出 neutral 类吗? 我想未必, 我们更可能关注 negative 或者 positive 这种情绪倾向明显的. 因此我认为现在这种平衡已经足够.

4.4 降维可视化

对特征进行降维可视化可以有效地帮助我们观察模型到底学到了什么. 下面两张图是对拼接模型的融合特征向量以及经过一层 MLP 后的向量的 PCA/T-SNE/UMAP 降维. 可以看到无论是哪种降维方法, 一开始的融合特征向量基本是完全杂乱五章的分布:

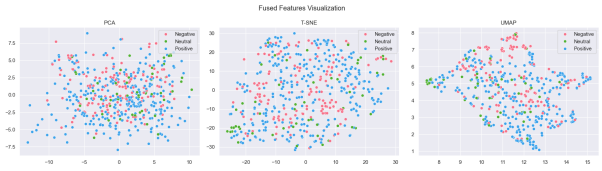

Figure 12 Fused Features, Concat

但 MLP 的隐层向量则是在 positive 和 negative 类 (蓝点和红点) 中有较好的分离度, 这印证了这一层 MLP 的必要性; 同时我们可以看到 neutral 类 (绿点) 仍然是散乱分布, 这和我们之前发现的模型对 neutral 类分类能力差是相符的.

Figure 13 Hidden Features, Concat

此外我们还可以发现, 对于注意力的 Attention_Alt, 他的融合向量已经有了较好的分离度, 所以没必要再引入 MLP.

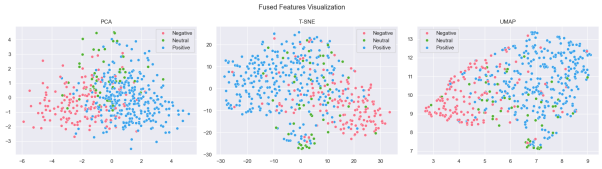

Figure 14 Fused Features, Attention_Alt

4.5 可解释性探索

由于我能力有限, 这部分仅是简单探索. 我尝试了用 GradCAM 方法分析模型图片数据的可解释性, 即基于交叉熵损失, 通过 ViT 的 patch embedding(conv1) 层的梯度计算 CAM, 示例如下:

Figure 15 GradCAM

这个结果看上去比较奇怪. 在错误案例分析中我们就发现对于这个数据集的分类任务而言, 如果光看图片的话, 我们人类也很难理解, 例如这个图的标签是 positive, 但实在是很难分析它和「情绪积极」有什么联系.

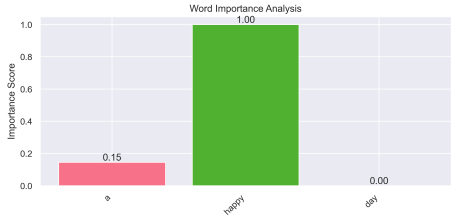

Figure 16 Text Importance

用类似的思路可以分析文本输入的特征重要性, 以 token_embedding 层为目标, 用「a happy day」做测试输入 (目标 positive 类), 发现 happy 的 GradCAM 权重最高, 符合预期.

5 拾遗

实验中遇到的一个问题是所提供的 txt 并没有统一编码, 因此我先通过一个预处理代码检测所有 txt 文件的编码并统一转化成 UTF-8 编码. 在构建 Dataset 的时候, 由于 CLIP 的 tokenize 默认只支持最长 77, 实验中部分文本超出了这个限制, 只能进行截断. 我们还可以考虑对输入做进一步预处理, 如对文本过滤掉无意义的符号 (@,# 等), 甚至可以考虑先都翻译成英语 (预训练模型可能没有在这些语言上训练过), 但这不是本次实验的主线.

对于测试集, 鉴于我们在实验中训练了三种不同融合方式的模型, 它们总体处于一个水平, 所以我借鉴了集成学习的思想, 用这三个模型分别预测, 然后进行多数投票得到最终结果. 对于三个模型结果都不一样的场景, 我将其设为了 neutral, 因为从图11中我们看到 neutral 类是最容易出现模型预测结果混淆的.