模式识别大作业 报告

一．实现方案

在实现方案方面，本次作业实际上尝试了两种方案，第一种方案是使用 embedding 和 ResNet 分别将标签和模型转化为向量，对比两个向量的差异，然后改进网络。第二种方案使用现成的图像文字匹配算法 clip，它的原理也是使用两个网络分别将文字和图片转化成向量然后使用两个向量的偏差分别对两个网络进行反向传播和优化。

报告中会介绍两种方案的具体实现方法，但是由于第一种方案因为一些原因（后文会详细说明）而效果非常差，因此关于这一方案仅介绍其实现方法而不介绍其训练和运行结果。作业提交的运行结果由第二种方法产生。

1. 第一种方案

1.1.embedding

首先使用 embedding 将文字转化成向量，但是考虑到自己训练 embedding 模型从数据获取到训练都十分困难，因此放弃了自己训练的方法，直接使用了现成的 embedding 数据：腾讯 AI Lab 发布的中文词和短语的 embedding[1]，其中有数个版本，我们使用的是 2000000 词，100 维向量的版本，经过搜索发现绝大部分表示颜色的词和短语都包含在其中。

在使用之前，必须从所有的词语中抽取出表示颜色的词语，但是由于其保存词语-向量映射的 txt 文件太大，实际使用中使用 txt 分割软件将其分成了 20 份，然后挨个查找其中包含颜色的词语，将词语和向量保存在一个新的 txt 文件中。

关于如何在其中查找颜色词语，需要首先用后文介绍的方法得到训练集中所有商品中提取的标准颜色标签词语，然后查找。

1.2. 标准化商品的颜色标签

使用如下的流程从商品标签中分离出比较标准的颜色标签。

首先根据大中小括号和加号截断字符串，取第一部分，后面的舍弃。然后挨个判断字符串每个字符的 asc 码，asc 码范围在 0-128 范围内的字符被删除，这样就去除了所有的英文字母、数字、标点和其他的符号，得到了一个较为标准的颜色标签。

然后，在刚才的 embedding 中搜索有无得到的颜色标签，如果有，那么按照 1.1 中的步骤，保存这个词语和它的向量；如果没有，那么在这个字符串中寻找有没有标准颜色库中的字（比如红、黄等），如果有的话这个图片的颜色标签就是这个标准颜色，如果没有的话就无法判断颜色标签了，那么对应的向量就是全零。

1.3. 代码实现

此部分的代码实现没有包含在提交的作业中，提交的作业的代码是第二种方案的，但是第二种方案的数据处理部分的代码绝大多数继承自第一部分且大部分相同。

数据处理部分实现了两个类：Tokenizer 和 mydataset

Tokenizer 类用于保存颜色标签和向量的对应关系，其中的函数有：

get_standard_tag_from_json:此函数用于读取训练集的 json 文件，从中提取所有的颜色标签，将其标准化，得到一个所有的标准化颜色标签构成的列表。

get_color_emb_table:得到训练集中所有的标准颜色标签之后，在下载的腾讯 embedding 数据中查找，将查找到的词语以及它的向量存入 txt 文件。

[1] https://ai.tencent.com/ailab/nlp/zh/embedding.html

generate_standard_tag_txt:将获得的所有标准颜色标签存入一个 txt 文件。

__call__:输入颜色标签名称，输出它对应的向量。

Mydataset 类用于存储和读取训练集的数据。在前面的 tokenizer 已经完成构建的情况下，mydataset 读取每一个训练集图片以及对应，使用 tokenizer 将标签转化为向量，就构成了将图片转化为向量的模型的输入和输出，就可以进行训练。

总结：tokenizer 首先读取所有的训练集颜色标签，将其标准化，然后按照这些标准化标签搜索腾讯 embedding 数据中对应词语的向量，然后将这些词语和对应的向量保存成一个单独的 txt 文件。然后 mydataset 读取数据时，先读取图片，然后使用 tokenizer 将图片对应的标签先转化成标准颜色标签再转化成向量，就可以训练了。

之所以要先搜集所有的训练集图片标签，再将它们从腾讯 embedding 数据中提取出来单独保存，而不是直接在运行的时候直接在腾讯 embedding 数据中搜索，是因为腾讯 embedding 数据集实在太大了，我们使用的 2000000 词的数据集 txt 文件大小大于 700Mb，大部分文档编辑器无法打开，搜索过程也极为缓慢。

1.4. 此模型效果不好的原因

实践中发现此方案的效果非常差，Acc 和 EM 分别只有不到 50%和 10%的水平，原因主要有以下几点：

（1） 使用的现成的腾讯 embedding 数据是不合适的，因为首先此数据集是在比较泛用的语言环境下训练的，而非本作业下单一的描述颜色的语言环境，因此向量之间的距离并不能很好的表示词语意思的远近关系。并且，有不少词语是多义词，在 embedding 中的向量可能更接近它的另一种含义。

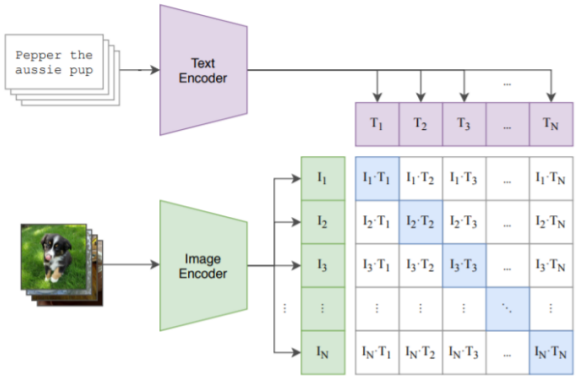
（2） 使用现成的 embedding 意味着从词语到向量的过程固定了，不可训练，模型中可训练的部分只有从图片到向量的部分，显然这会降低模型的泛化能力。

（3） 对颜色标签的标准化也有问题，某些标签标准化之后的词语并不直接表示颜色，例如“青苹果”，“摩卡”等，这些词语在 embedding 中的向量显然就更接近其原本的含义而不是表示颜色的含义。

2. 第二种方案

2.1.clip

Clip 模型分为两个模型，图像编码器和文本编码器，图像编码器可以说 resnet，文本编码器可以是 transformer。



如下图：对于 n 张图片和 n 个文字描述，图像编码器和文本编码器分别得到 n 个向量，目标是最大化同一对图像和文本的向量内积，即矩阵对角线上的元素。训练时分别计算对角

线元素对于文字向量和图片向量的梯度，分别对两个模型进行反向传播。

2.2. 中文转英文

因为现有的 clip 模型是按照英文文字描述训练的，因此需要把标签转为英文。因为在方案一中，我们已经得到了训练集中所有的颜色标签的标准化并且将其保存为了 txt，因此这里直接用有道文档翻译把这个 txt 转成了英文，与原来的标准化中文颜色标签一一对应。

2.3.代码实现

数据处理和读取方面仍然是两个类：Tokenizer 和 mydataset

Tokenizer 类与方案一中几乎相同，只不过保存的标签的向量变成了标签的英文翻译。函数：

build_tag_english_dict：读取前面保存的所有标准颜色标签的 txt 文件，以及它的英文翻译文档，构建一个 dict，保存中文-英文对应关系。

__call__：输入中文标签，输出英文标签。

Mydataset：保存所有的图片的列表，每张图片对应的中英文标签（训练集），每张图片对应的可选标签的中英文（测试集）。读取数据时，如果是训练集，返回图片以及英文文字描述，如果是测试集，返回图片以及英文的可选标签。

数据处理完成后，需要安装 clip，安装方法参考了 GitHub 上的 clip 项目[2]。

2.3.1.测试代码

测试时首先需要加载模型，如果直接使用 clip 项目自己的模型，一共有 8 种；如果使用自己的模型，需要首先加载 clip 的"ViT-B/32"模型，然后把自己的模型的 model_state_dict 加载到它。

测试时，包括训练时也一样，输入模型的文字描述并不是刚才得到的标准颜色标签，而是一个短语：“A photo of a xxx color cloth”，如果图片的文字标签是 red，那么输入模型的文字描述是“A photo of a red color cloth”，因为 github 上的作者说这样有助于提高准确度。

2.3.2.训练代码

此部分代码完全参考了 github 上 clip 项目中一篇 issue 中的代码[3]。

训练在 clip 的"ViT-B/32"模型的基础上进行。

训练时需要同时训练图片编码器和文本编码器，因此需要两个损失函数，分别计算图片向量和文本向量与 ground_truth 的误差。

训练时使用 Adam 优化器，参数设置为 lr=5e-5，betas=(0.9,0.98)，eps=1e-6，weight_decay
=0.2，后来发现 weight_decay 过大导致收敛速度变慢，因此后来 weight_decay 改为 0.02。

二．总结

本次作业尝试了两种方法，第一种方法事实上只有一个图片编码器有优化的空间，而文本编码器因为使用固定的 embedding 数据而不能很好的抽取文本的特征，不能与图片编码器配合减小误差。因此效果很差。

[2] https://github.com/openai/CLIP

[3] https://github.com/openai/CLIP/issues/83

第二种方法使用了 clip，其同时具有图片编码器和文本编码器，能够同时训练，因此能够很好的同时抽取文本和图片的特征，获得了较好的效果。

项目中的问题和可能改进的方面有以下几点：

（1）原始标签的处理：

事实上我们实现的原始标签的处理比较简单，即简单的分割以及去除非法字符，然后查看它是否在 embedding 数据集中存在，如果不存在就寻找标签中的标准色库文字。这种处理的泛用性不高，遇到比较不规则的标签时就会失效。可能的改进方法是，对于一个商品的多幅图片和多个标签，可以考虑去除这些标签中相同的部分，例如：


```
"623031615199": {
    "optional_tags": [
        "蓝色可乐小雏菊",
        "白色可乐小雏菊",
        "黑色可乐小雏菊"
    ],
```


如果去除标签中相同的部分，就可以比较好的去除干扰，这样几个标签联合处理的方法可能会优于单个单个的处理。

（2）模型选择问题

在 clip 的所有模型中，'RN50x64'，'ViT-L/14'这两个是最复杂，效果最好的，即使不训练也能达到 83%以上的 Acc 和 54%以上的 EM，但是因为太复杂，所以即使是使用 kaggle 的服务器训练使用最小尺寸的数据集也非常慢（一个 epoch 可能要花费数小时），并且经常显存溢出，如果有更好的计算资源可能可以尝试这两个模型的效果。