MovieLens 期末大作业

2xxxxxxxxx

June 2024

1 数据处理与数据集搭建

1.1 数据筛选

部分电影没有对应海报，所以要取出 info.csv 和 poster 文件夹中的交集，得到最终电影数量 $N = 2938$，得到 `df_movie_info` 如下。

Table 1: `df_movie_info.head(3)`

id	name	intro
1	Toy Story (1995)	A cowboy doll is profoundly threatened and jea...
2	Jumanji (1995)	When two kids find and play a magical board ga...
3	Grumpier Old Men (1995)	John and Max resolve to save their beloved bai...

1.2 数据预处理

1.2.1 对文本进行预处理

对电影简介预处理的方式主要包括：

1. 把文字切片后删除掉文本内的标点符号和空白，并把所有文本切换成小写
2. 使用 HW09 的 STOPWORDS 除去文本里无意义的停词
3. 使用 WordNetLemmatizer 将单词转换为原本的形态

1.2.2 对电影进行预处理

本部分没有对图像进行处理，只是搜集齐了所有电影海报的路径。

1.3 数据集搭建

根据题目要求，根据 train.txt 内的数据认为是训练集，所有数据样本认为是测试集。

2 模型设计与训练

2.1 模型设计

2.1.1 搭建 Dataset 和 Dataloader

Dataset 当中应当包含 poster 和 intro 的信息，其中 intro 的文本预处理已经在上一步进行完成，这一步不做处理，直接将 string 类型数据放入下一步的 TextModel 里处理。

这里主要处理 poster 信息,我们设计了一个 `preprocess_poster(self,idx)` 函数，用于读取图片，并对图片进行 resize。核心函数代码如下：

```
class MovieLensDataset(Dataset):
    def __init__(self, data_root, posters, intros,
        **kwargs):
        self.data_root = data_root
        self.posters = posters
        self.intros = intros
        self.transform = transforms.Compose([
            transforms.ToPILImage(),
            transforms.Resize((224, 224)),
            transforms.ToTensor(),])

    def __len__(self):
        return len(self.posters)

    def preprocess_poster(self,idx):
        poster_path = os.path.join(self.data_root,
            self.posters[idx])
        poster = cv2.imread(poster_path)
```

```
17          poster = cv2.cvtColor(poster, cv2.
                COLOR_BGR2RGB)
18          poster = self.transform(poster)
19          return poster
20
21      def __getitem__(self, index):
22          poster = self.preprocess_poster(index)
23          intro = self.intros[index]
24          return poster, intro
```

搭建完成 Dataset 后我们创建 Dataloader，并设定 $batch\_size = 32$, $num\_workers = 4$。

2.1.2 设计 PosterModel 和 TextModel

分别使用 Resnet50 和 DistilBert 模型

针对 PosterModel 首先对图像数据进行归一化处理，再通过 Resnet50 的模型，并修改输出层。

针对 TextModel 模型，首先对它做 Token 化处理，再通过 DistilBert 模型获得输出。代码如下：

```
1  from transformers import DistilBertTokenizer,
       DistilBertModel
2
3  class PosterModel(nn.Module):
4      def __init__(self):
5          super(PosterModel, self).__init__()
6          self.resnet = models.resnet50(pretrained=
               True).to(device)
7          self.resnet.fc = nn.Identity()
8          self.transform = transforms.Compose([
               transforms.Normalize(mean=[0.485, 0.456,
               0.406], std=[0.229, 0.224, 0.225])])
9
10     def forward(self, image_tensor):
11         image_tensor = self.transform(image_tensor)
12         self.resnet.eval()
13         with torch.no_grad():
14             outputs = self.resnet(image_tensor)
15         return outputs
```

```
16
17 class TextModel(nn.Module):
18     def __init__(self):
19         super(TextModel, self).__init__()
20         self.token = DistilBertTokenizer.
               from_pretrained('distilbert-base-uncased'
               )
21         self.model = DistilBertModel.from_pretrained
               ('distilbert-base-uncased').to(device)
22
23     def forward(self, x):
24         text_in = self.token(x, return_tensors="pt",
                padding=True, truncation=True,
               max_length=512).to(device)
25         text_out= self.model(**text_in)
26         text_feat= text_out.last_hidden_state[:, 0,
               :]
27         return text_feat
```

2.1.3 创建 CombinedModel

把 PosterModel 和 TextModel 结合起来，并且分别针对 poster 和 intro 设置多层感知机，最后获得处理好的海报和文本特征。

其中，多层感知机当中都包含了三个隐藏层，每两个隐藏层之间都添加了批标准化层和 ReLU 激活函数，以及用于防止过拟合的 Dropout 层。

2.2 训练流程

2.2.1 损失函数设计

通过计算图像特征和文本特征之间的相似性（用余弦相似度来衡量），用温度参数减弱相似性的强度，使用交叉熵损失函数分别计算图像和文本的损失，最终的对比损失是图像损失和文本损失的平均值。代码如下：

```
1 def contrastive_loss(img_features, text_features,
      temperature=0.05):
2     cosine_similarity = nn.CosineSimilarity(dim=-1)
3     img_features = img_features.unsqueeze(1)
4     text_features = text_features.unsqueeze(0)
```

```
    similarities = cosine_similarity(img_features,
        text_features)
    logits = similarities / temperature
    labels = torch.arange(len(img_features)).to(
        logits.device)

    loss_img = nn.CrossEntropyLoss()(logits, labels)
    loss_text = nn.CrossEntropyLoss()(logits.T,
        labels)
    return (loss_img + loss_text) / 2
```

2.2.2 其他设计

使用 Adam 优化器（参数包括 0.001 的学习率）和 StepLR 学习率调度器

同时利用混合精度训练，来提高训练效率

2.3 模型延展设计-CLIP

通过”openai/clip-vit-base-patch32” 直接使用 CLIP 模型，将上文的 PosterModel 和 TextModel 替换成 CLIPModel 和 CLIPProcessor 的对应内容，其余内容包括多层感知机和损失函数、优化器等和前文一致。

3 测试与分析

3.1 评测结果

首先我们的距离函数定义为 1-余弦相似性，代码如下：

```
def get_dist_matrix(model, dataloader):
    model.eval()
    img_features = []
    text_features = []
    with torch.no_grad():
        for poster, intro in tqdm(dataloader, total=
            len(dataloader)):
             poster = poster.to(device)
```

```
            img_feature, text_feature = model(poster
                , intro)
            img_features.extend(img_feature)
            text_features.extend(text_feature)

    img_features_tensor = torch.stack(img_features)
    text_features_tensor = torch.stack(text_features
        )

    cosine_similarity = torch.mm(img_features_tensor
        , text_features_tensor.T)
    img_norm = torch.norm(img_features_tensor, dim
        =1, keepdim=True)
    text_norm = torch.norm(text_features_tensor, dim
        =1, keepdim=True)

    distances = 1 - cosine_similarity / (img_norm *
        text_norm.T)

    return distances
```

根据 get_acc 函数得到评测结果如下：

Table 2: Accuracy

dim	原模型	CLIP
0	65.66%	62.87%
1	66.30%	62.4%

3.2 延展分析

3.2.1 原模型和 CLIP 的准确率差异

CLIP 的效果不如原模型好，是因为 CLIP 在进入 MLP 之前的输出维度为 512，而原模型中 Resnet50 的输出维度为 2048，DistilBert 的输出维度为 768，原模型比 CLIP 有更多的信息，所以效果更好。

3.2.2 选取一些海报/简介，利用你所得到的距离矩阵 D 挑选出与之距离最接近的 top-k（k 可自选）简介/海报

我们随机选取两个图片和对应的前五个简介描述。

第一个（图片见 Figure1）：

1. self destructing world vengeful australian policeman set stop violent motorcycle gang
2. bumbling cadet street inept goon successfully orchestrate metropolitan crime wave
3. basketball superstar dennis rodman star hip interpol agent attempting defeat deadly plan crazed arm dealer
4. brother dy mysterious circumstance car accident london gangster jack carter travel newcastle investigate
5. professional hit man robert rath want fulfill contract retiring unscrupulous ambitious newcomer hit man miguel bain keep killing rath target

Figure 1: Example 1

第二个（图片见 Figure2）：

1. convict pose cop retrieve diamond stole year ago
2. ex used police lure criminal hiding
3. australian outback expert protects new york love gangster ve followed
4. cousin unknowingly rob mob face dangerous consequence
5. retired master car thief come industry steal car crew night save brother life

Figure 2: Example 2

我们随机选取两个简介和对应的前五个海报。

第一个（图片见 Figure3）:

obese attorney cursed gypsy rapidly uncontrollably lose weight

第二个（图片见 Figure4）:

secret service agent frank horrigan clint eastwood couldn save kennedy determined let clever assassin president

回答问题:

1. 在这一数据集上电影海报与电影简介内容是否具有某些相关性?

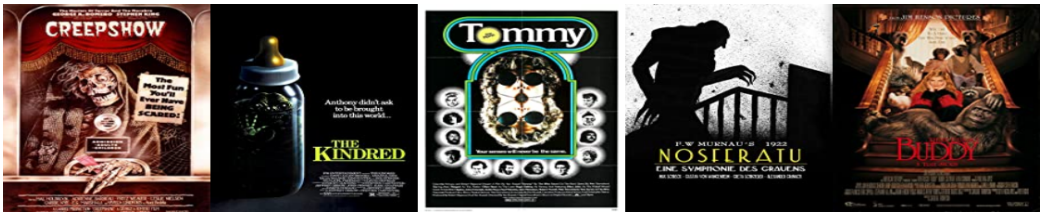

Figure 3: Example 1

Figure 4: Example 2

电影海报和电影内容大部分时候有相关性但不一定。比如第一个示例 Party Girl，其主要内容为一个女孩被保释出狱后开始工作改变人生，而海报中仅有一个时尚的女郎形象。

但第二个示例 Blue Streak 就相关性强烈，该电影主要就是讲述窃贼为了拿到自己之前藏起来的钻石的故事，电影中的主角举着一块钻石，这就具有相关性。

2. 对于给定的海报/简介，最接近的前 K 个简介/海报是否共享相似的语义内容、主题、风格、或其他特征?

是的。当选取海报看前五个简介时，能发现 Party Girl 筛选出来的内容都与犯罪相关，比如 policeman，arm dealer，gangster，crime；Blue Streak 的内容也都与犯罪相关，比如 cop，stole,gangster,rob,thief。

当选取简介看前五个海报的时候，发现第一组都是恐怖风格，第二组都有穿西装的男人。

3. 跨模态对齐的特征可能在哪些下游任务上发挥怎么样的作用?

用于智能检索，比如 AI 识别物品，在这个任务中就是海报图片识别电影内容，进一步的也可以是电影截图识别电影标题等等。