

当代人工智能实验报告 5

温兆和 10205501432

2024 年 1 月 6 日

1 实验目的

在本次实验中，我们使用 PyTorch 工具构建并训练了一个多模态模型，使得模型能够同时根据文字和图片对各种信息进行情感分析。

您可以在 [GitHub 访问这个项目](#)。本门课程的 GitHub 仓库的 visibility 将会在本次实验截止日期抵达后转为 public。

2 实验环境

由于构建的模型并不复杂，本次实验是在本地进行的。需要安装 Python 3.10。

需要安装的工具包有：

- numpy
- torch
- pandas
- PILLOW
- tensorflow
- torchvision
- sklearn

如果需要安装这些包，可以在项目路径下执行 `pip install -r requirements.txt` 命令。

3 实验步骤

3.1 数据预处理

3.1.1 图像处理

在本次实验中，各条数据中的图片大小不一，但神经网络的输入都必须是相同维度的张量。所以，我们需要用 PyTorch 中的 Transform 工具把所有图片都转化为大小相等的张量。基于本次实验选用的模型，我们把图片的尺寸转化为 $224 * 224 * 3$ 。

```
from torchvision import transforms
2
class MyDataset(Dataset):
4     def __init__(self, dataframe, transform):
        ...
6         self.transform = transform
```

```

8      ... ..
10     def __getitem__(self, index):
11         img_data = self.data.iloc[index]['fig']
12         txt_data = self.data.iloc[index]['docvec']
13         label = self.data.iloc[index]['label']
14
15         if self.transform:
16             img_data = self.transform(img_data)
17
18         return img_data, txt_data, label

```

3.1.2 文字处理

在文字处理中，我们使用 TensorFlow 中的 Tokenizer 工具把所有的文字转化为大小不等的向量，再将它们填充为指定长度。

```

from tensorflow.keras.preprocessing.text import Tokenizer
2 from tensorflow.keras.preprocessing.sequence import pad_sequences

4 combined_corpus = pd.concat([train_and_valid['txt'], test['txt']], axis=0)
max_len = 50
6 tokenizer = Tokenizer()
tokenizer.fit_on_texts(combined_corpus)
8 train_seq = tokenizer.texts_to_sequences(train_and_valid['txt'])
test_seq = tokenizer.texts_to_sequences(test['txt'])
10 train_seq = pad_sequences(train_seq, maxlen=max_len, padding='post')
test_seq = pad_sequences(test_seq, maxlen=max_len, padding='post')

```

3.2 模型原理与构建

3.2.1 图片处理模型 AlexNet

在第三次实验中，AlexNet 是一种正确率较高且结构较简单、运行时间较短的神经网络。因此，我们在本次实验中，我们仍然使用 AlexNet 来对图像进行处理。与上次不同的是，由于本次实验中的图片是彩色图片而不是黑白图片，第一个卷积层中的通道数设置为 3。此外，输出的维度也未必是类别数，接下去我们还要结合文字数据进一步处理。

```

1 class AlexNet(nn.Module):
2     def __init__(self, outputdim, input_channels=3):
3         super(AlexNet, self).__init__()
4         self.features = nn.Sequential(
5             nn.Conv2d(input_channels, 96, kernel_size=11, stride=4, padding=2),
6             nn.ReLU(inplace=True),
7             nn.MaxPool2d(kernel_size=3, stride=2),
8             nn.Conv2d(96, 256, kernel_size=5, padding=2),
9             nn.ReLU(inplace=True),
10            nn.MaxPool2d(kernel_size=3, stride=2),
11            nn.Conv2d(256, 384, kernel_size=3, padding=1),
12            nn.ReLU(inplace=True),

```

```

13         nn.Conv2d(384, 384, kernel_size=3, padding=1),
14         nn.ReLU(inplace=True),
15         nn.Conv2d(384, 256, kernel_size=3, padding=1),
16         nn.ReLU(inplace=True),
17         nn.MaxPool2d(kernel_size=3, stride=2),
18     )
19     self.classifier = nn.Sequential(
20         nn.Dropout(),
21         nn.Linear(256 * 6 * 6, 4096),
22         nn.ReLU(inplace=True),
23         nn.Dropout(),
24         nn.Linear(4096, 4096),
25         nn.ReLU(inplace=True),
26         nn.Linear(4096, outputdim),
27     )
28
29     def forward(self, x):
30         x = self.features(x)
31         x = x.view(x.size(0), 256 * 6 * 6)
32         x = self.classifier(x)
33     return x

```

3.2.2 文字处理模型 TextRNN

在第一次实验中 TextRNN 模型在一个十分类的数据集上的准确率达到了 99%，而且其结构简单，训练时间较短，所以本次实验也还是沿用实验一中的 TextRNN 模型对文字进行处理。这个网络中使用了 LSTM，它能解决传统 RNN 中的梯度消失和梯度爆炸问题。

```

1 class TextRNN(nn.Module):
2     def __init__(self, vocab_size, embedding_dim, hidden_size, outputdim):
3         super(TextRNN, self).__init__()
4         self.embedding = nn.Embedding(vocab_size, embedding_dim)
5         self.lstm = nn.LSTM(embedding_dim, hidden_size, batch_first=True)
6         self.fc = nn.Linear(hidden_size, outputdim)
7         self.softmax = nn.Softmax(dim=1)
8
9     def forward(self, x):
10         embedded = self.embedding(x)
11         lstm_out, _ = self.lstm(embedded)
12         lstm_out = lstm_out[:, -1, :]
13         output = self.fc(lstm_out)
14         output = self.softmax(output)
15     return output

```

3.2.3 多模态综合处理模型 MultiModel

在多模态情感分类任务中，我们需要综合文字和图片数据来分析某条数据对应的情感。我们把图像数据和文本数据分别输入上面的 AlexNet 和 TextRNN，会得到两个输出。我们把这两个输出拼接起来，经过一个 ReLU 层和两个线性层，最终得到一个三维的输出：

```

1 class MultiModel(nn.Module):
2     def __init__(self, vocab_size, embedding_dim, single_outputdim, hidden_size, num_classes):
3         super(MultiModel, self).__init__()
4         self.img_model = AlexNet.AlexNet(single_outputdim, 3)
5         self.txt_model = TextRNN.TextRNN(vocab_size, embedding_dim, hidden_size, single_outputdim)
6         self.act = nn.ReLU(inplace=True)
7         self.fc1 = nn.Linear(single_outputdim * 2, single_outputdim)
8         self.fc2 = nn.Linear(single_outputdim, num_classes)
9
10    def forward(self, img_data, txt_data, mode):
11        img_out = self.img_model(img_data)
12        txt_out = self.txt_model(txt_data)
13        ...
14        concatenated_out = torch.concat((img_out, txt_out), dim=1)
15        out = self.fc1(self.act(concatated_out))
16        final_out = self.fc2(out)
17        return final_out

```

3.3 模型训练与测试

我们用交叉熵损失作为损失函数，使用 Adam 优化器，把学习率设置为 0.001，训练十个 epoch，得到如下结果。

3.3.1 融合模型结果

总体来说，这个模型能在验证集上达到 60% 以上的准确率。但是有一个比较反常的现象，就

```

Epoch 1/10, Training Loss: 0.7468326687812885, Validation Loss: 1.1092476844787598, Accuracy: 0.60375
Epoch 2/10, Training Loss: 0.9586592154502869, Validation Loss: 1.0439907312393188, Accuracy: 0.60375
Epoch 3/10, Training Loss: 0.8410295844078864, Validation Loss: 1.0632294416427612, Accuracy: 0.60375
Epoch 4/10, Training Loss: 1.1215787863674927, Validation Loss: 1.0337879763412476, Accuracy: 0.60375
Epoch 5/10, Training Loss: 0.9234484424127880, Validation Loss: 1.0248923455892304, Accuracy: 0.60375
Epoch 6/10, Training Loss: 0.9642479419788252, Validation Loss: 1.0443671941757262, Accuracy: 0.60375
Epoch 7/10, Training Loss: 0.8611043691635132, Validation Loss: 1.0279574394226874, Accuracy: 0.60375
Epoch 8/10, Training Loss: 0.9311248668087585, Validation Loss: 1.0236858235168457, Accuracy: 0.60375
Epoch 9/10, Training Loss: 0.9978228423789970, Validation Loss: 0.9879455241866248, Accuracy: 0.60375
Epoch 10/10, Training Loss: 0.9390528282856885, Validation Loss: 1.0351219448874512, Accuracy: 0.6025

```

是无论损失函数的值如何变化，模型的准确率从第一个 epoch 开始就稳定在 60.375%，没有再上升。我试着增加了卷积层的层数，并把 ReLU 函数更换为 tanh、ELU 等激活函数，但仍然没能解决这个问题，便只能作罢。我还试着把学习率跳到很小 $1e-307$ 再训练模型，这时每一个 epoch 的验证准确率就会稳定在 10% 到 20% 这样一个比较低的水准。所以我推测，可能模型在训练完第一个 epoch 的时候就达到了损失函数的最优值点。我还打印出了模型在验证集上的预测结果，发现里面几乎全是 positive。这可能还是由于模型还是比较简单。而且，训练集中 positive 的样本的数量也占到了大多数，所以模型也很难学习到 neutral 和 negative 样本的足够知识。

3.3.2 消融实验结果

消融实验就是分别只输入文本或图像数据，查看模型的表现。如果只输入图像，那就把文本模型的输出全部置为零，反之亦然：

```

1     if mode == 1:         # 仅输入文本
2         img_out *= 0
3     elif mode == 2:       # 仅输入图像
4         txt_out *= 0

```

我们得到的结果如下：这是一个耐人寻味的结果，只输入图像就能得到与融合实验差不多的准确

```
Text Only Loss: 1.0718899965286255, Accuracy: 0.2925  
Image Only Loss: 1.041540265083313, Accuracy: 0.595
```

率，而只输入文本，模型表现就会差很多。看来在融合实验中，也是图像起了主要作用。

4 总结

本次实验中，我们综合运用计算机视觉和自然语言处理的知识，通过图像、语言等不同形态的数据分析当事人的情感，更加深入地体会了人工智能的强大。