方面级别情感分类实验报告

人工智能学院 周韧哲 181220076

目录树

```
HW2
├── data
│   ├── result
│   │   └── 181220076.txt    #分类结果
│   ├── test.txt
│   └── train.txt
├── model.py    #BertModel class
├── utils.py    #Data Process
└── main.py     #main loop
```

运行

`Requirements: python==3.6.9, pytorch==1.7.0, cuda==10.1, transformers==3.5.1, numpy==1.18.5`

我在google colaboratory运行的这份代码，其GPU使用的是Tesla T4，由于Tesla显卡和英伟达显卡架构不同，同样的代码运行效果可能不同，如**无法复现榜单结果，请与我联系**。

一些重要的命令行参数有：

```
optional arguments:
  --data_path      data path
  --num_iters      iter nums of outer loop
  --seed           seed
  --batch_size     batch size per minibatch
  --dropout        dropout prob in final layer
  --max_len        max lenth of word vector
  --lr             learning rate
  --weight_decay   l2 regulation rate
  --hidden_dropout_prob           hidden dropout prob in pretrained bert
model
  --attention_probs_dropout_prob  attention dropout prob in pretrained bert
model
```

命令行键入`python main.py`会用默认参数运行代码，结果保存在/data/result/181220076.txt中，即为榜单结果。

具体实现

数据处理

- 我使用了bert预训练模型。因为我们的输入有句子sentence与方面词aspect，所以容易构建bert的输入为：

$$[\text{CLS}]\ \text{sentence}\ [\text{SEP}]\ \text{aspect}\ [\text{SEP}]$$

如对与train.txt中的

```
Avoid this $T$ !
place
```

我将其转换为

```
[CLS] Avoid this place ! [SEP] place [SEP]
```

并使用BertTokenizer将其转换为词向量input_ids，由于输入为两句话，再构建token_type_ids，并将其标准化为固定长度。详细请见utils/DataBuffer。

模型构建

- 预训练模型为bert-base-uncased，首先把句子输入到Bert预训练模型，然后将句子的embedding输入给一个dropout层后经过一层全连接层输出。

```
def forward(self, input_ids, token_type_ids):
    _, output = self.bert(input_ids,token_type_ids)
    output = self.dropout(output)
    output = self.dense(output)
    return output
```

输出的维度为3，代表该输入在3个label上的概率。详见model/Bert_Model。

训练过程

- 使用Adam优化器，交叉熵损失函数，训练num_iters轮，每轮内进行minibatch更新：

```
for i_iter in range(args.num_iters):
    n = math.ceil(len(data_buffer)/args.batch_size)
    for i in range(n):
        input_ids,token_type_ids,label = data_buffer.get_train_data()
        optimizer.zero_grad()
        output = bert_model(input_ids, token_type_ids)
        loss = criterion(output, label)
        loss.backward()
        optimizer.step()
```

详见main/main_loop()。

实验总结

- 经过调参，在测试集上的最优分类准确率约为0.9061，将最优参数设为了默认参数。bert使用了大量语料训练，只需要对预训练模型进行fine-tuning就可以在此任务上获得优秀的结果，可见bert之强大。