Task 2 报告

1 数据预处理

代码详见repo中seq_cnn目录下的data_preprocess.py。

首先，先读取train.txt中的数据，再使用jieba对train.txt中的中文句子进行分词，并根据训练集的分词结果构造模型的词汇库，词汇库保存为vocab.json。然后利用jieba将train.txt、dev.txt、test.txt中的中文句子进行分词并用此前得到的词汇库将分出的token转化为词汇库中相应的序号，再将处理好的序列和标签成对匹配后分别保存为train.json、dev.json、test.json。

2 CNN参数

代码详见repo中seq_cnn目录下的model.py。

```
class CNN_rand(nn.Module):
    def __init__(self, config=CNNConfig()):
        super(CNN_rand, self).__init__()

        self.embedding = nn.Embedding(config.vocab_size, config.n_embd)
        self.conv_layer = nn.Conv2d(in_channels=1, out_channels=32,
                                    kernel_size=(3, config.n_embd),
                                    padding=(config.min_seq_len // 2, 0))
        self.linear = nn.Linear(32, config.n_class)

    def forward(self, x):
        x = self.embedding(x)[:, None, :, :]
        x = self.conv_layer(x)
        B, C, H, _ = x.shape
        x = x.reshape(B, C, H)
        x, _ = torch.max(x, dim=-1)
        x = self.linear(x)
        return x
```

模型由三部分构成：一个随机初始化的embedding layer、一个卷积层、一个线性层。其中config.vocab_size为11705、config.n_embd为256、config.n_class为4、config.min_seq_len=10。

计算过程和原来论文类似，也就是在embedding之后对序列的embedding做卷积，然后对卷积结果的各个channel分别取最大值作为线性层的输入，最终每个序列经过卷积和取最大值的结果经过线性层之后输出config.n_class个logits。在这个模型当中的话，在序列维度上卷积层的kernel大小为3，卷积层会输出32个channels（因此线性层是32*config.n_class的）。此外，因为有些序列长度过短，为了保证卷积能够进行，会在序列维度前后各zero-pad 5个长度为config.n_embd的零向量。

3 训练

代码详见repo中seq_cnn目录下的train.py。

batch size为100，optimizer使用的是Adagrad，optimizer相关参数使用的是Adagrad默认的参数，设置训练50个epoch，但是因为使用了early stopping所以不到5个epoch一般训练就结束了。

4 结果

在使用train.py训练时，在第4个epoch结束后early stopping被触发，最后在test set上面的accuracy为80.40%（结果可能会每次运行都不同，但是基本没见到过最后的Accuracy低于75%）。