PRML Assignment4 Report

Part 1

1、数据处理

Assignment4的任务是利用CNN与RNN完成文本分类的任务，与Assignment2相同利用sklearn中的fetch_20newsgroups获取所需数据，并且使用全部20个category的数据作为数据集。

首先，需要对数据进行清洗，去除数据中的数字、空白字符、标点符号等，在此基础上将所有字符转换为小写，并对文本进行切分split，即每一条数据对应于一个Word List，具体实现如下：

```
data_tmp = re.sub('\d+', ' ', data[i])
for c in string.whitespace:
    data_tmp = data_tmp.replace(c, ' ')
for c in string.punctuation:
    data_tmp = data_tmp.replace(c, '')
data_tmp = data_tmp.lower().split()
```

其次，利用FastNLP中提供的DataSet模块创建数据集合，并通过Instance向其中添加元素：

```
dataset = DataSet()
dataset.append(Instance(raw_sentence=data[i], target=int(target[i]), sentence=data_tmp)
```

以上操作能够得到初步的数据集，在此基础上进行以下的处理，得到最终需要的数据集：

```
dataset.apply(lambda x: x['raw_sentence'].lower(), new_field_name='raw_sentence')
dataset.apply(lambda x: len(x['sentence']), new_field_name='seq_len')
```

其中dataset中的每一条数据由以下部分组成：raw_sentence 初始文本，target 文本对应类别，sentence 文本切分后的Word List，seq_len 对应Word List的大小。

利用以上得到的dataset以及fastNLP.Vocabulary构建词典，如下方所示，先设置词频阈值min_freq为10（避免因为词频较小的词导致训练结果较差），利用训练集train_data生成所需的词典vocab；进而根据词典，在dataset中添加words词条，即words中每个int表示对应于sentence中每个词在词典中的index。

```
vocab = Vocabulary(min_freq=10).from_dataset(train_data, field_name='sentence')
vocab.index_dataset(train_data, field_name='sentence', new_field_name='words')
```

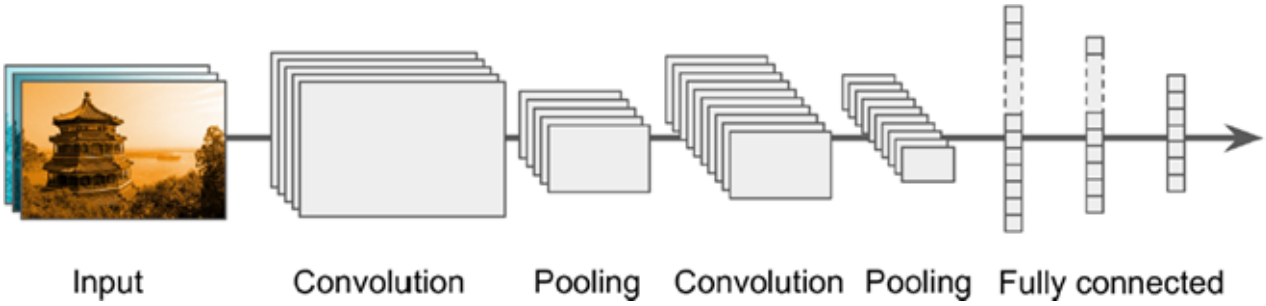
以上即为数据处理的部分。

2、CNN Model

- **模型实现**

  对于CNN分类器，首先谈谈对CNN文本分类的理解，其次阐述如何使用fastNLP加以实现：

  1、CNN的整体框架如下图所示，主要分为输入层、卷积层、池化层与全连接层。



  对于CNN文本分类，基本思想即为通过卷积核对embedding后的文章矩阵进行卷积，从而提取文章的特征，进而利用池化操作进行压缩，不断重复卷积-池化的操作，最终能得到一个特征向量，即可利用该特征向量进行线性分类任务。

  2、结合fastNLP，对于CNN的实现做说明，由于fastNLP已经封装了CNN所需的部分函数，因此在理解的基础上加以使用，具体如下：

  - 输入层中，利用fastNLP.encoder中封装的Embedding模块进行embedding，输入参数为元组tuple(vocab_size, embedding_dim)；
  - 卷积&池化层中，fastNLP.encoder中封装了模块ConvMaxpool，其中整体实现了卷积+池化的步骤。为了实现更好的文本分类效果，需要提取尽量多的文本特征，所以需要采用不同大小的多个卷积核对文本矩阵进行卷积，体现在参数out_channels、kernel_sizes中。其中kernel_sizes为不同大小的卷积核的集合，out_channels为各个不同大小卷积核的数量，与kernel_sizes中一一对应。在fastNLP中，池化层选择最大池化方法，将每个卷积核得到的特征向量池化为一个数，最终每个不同大小的卷积核都能得到对应out_channels大小的特征向量；
  - 全连接层中，采用nn.Linear，并且添加nn.dropout模块（防止过拟合），该层将上一层中得到的不同大小的卷积核得到的特征向量拼接起来，得到最终用于分类的向量。

- **模型参数与训练结果**

  1、参数表

  - **CNN**

embedding_dim	padding	dropout	out_channels	kernel_sizes
128	2	0.1	(4,5,6)	(3,4,5)

  - **Trainer**

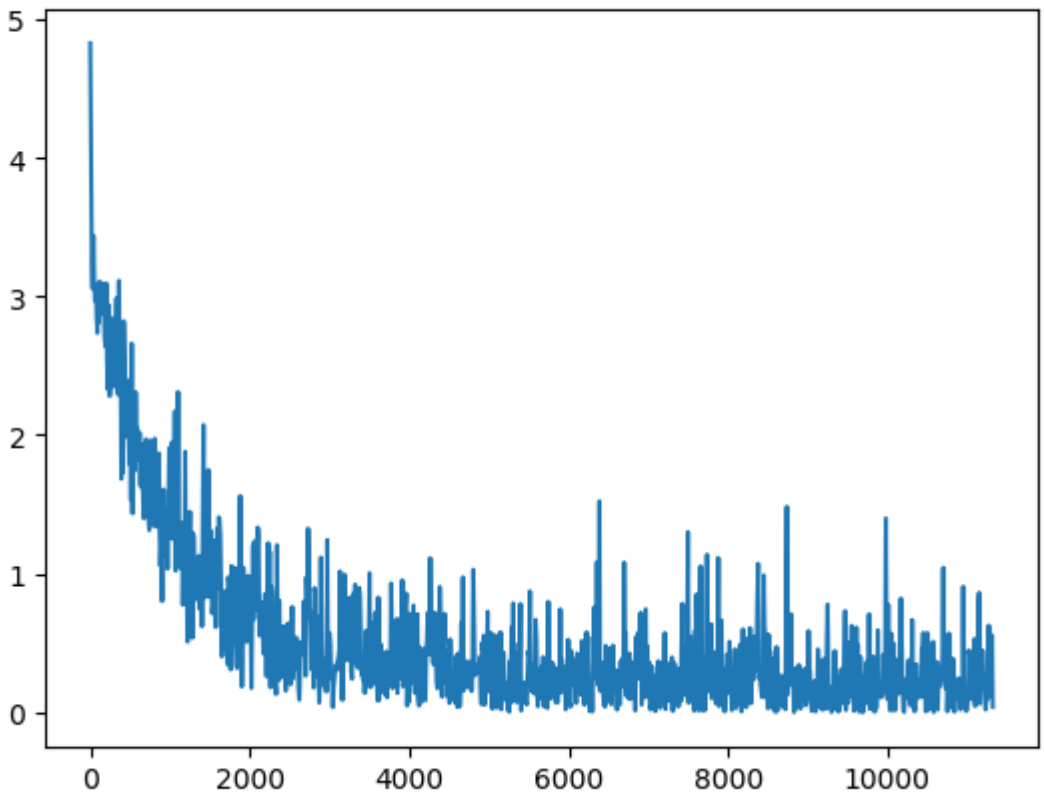
Loss	metrics	batch_size	n_epochs	device
CrossEntropyLoss	AccuracyMetric	16	20	cuda

2、训练结果

Accuracy	Loss
0.751105	0.18543009459972382

3、Loss曲线

（横坐标：iteration 纵坐标：loss）



3、RNN Model

- **模型实现**

  在该部分中，选择使用LSTM网络来实现RNN分类模型，基本思想为：

  - 首先输入词列表，同样利用fastNLP中的embedding将输入embedding为词向量；

- 其次通过LSTM层，利用fastNLP中封装的encoder.LSTM，输入序列通过网络的计算得到隐藏层的输出特征向量;
- 与CNN不同，在池化部分使用了均值池化Meanpolling进行池化操作;
- 对池化后的结果，利用nn.Linear（fc层）即可以得到分类预测

需要注意的是，在fastNLP的文档中，实现的LSTM中仅使用部分的网络输出作为分类预测的依据，在某些样本上由于提取的特征不足导致分类效果不好，因此选择整个序列在LSTM上的所有输出作为依据进行分类判别，保留了文章整体的特征信息。

- **模型参数与训练结果**

  1、参数表

  - **RNN**

embedding_dim	hidden_dim	num_layers
128	128	1

  - **Trainer**

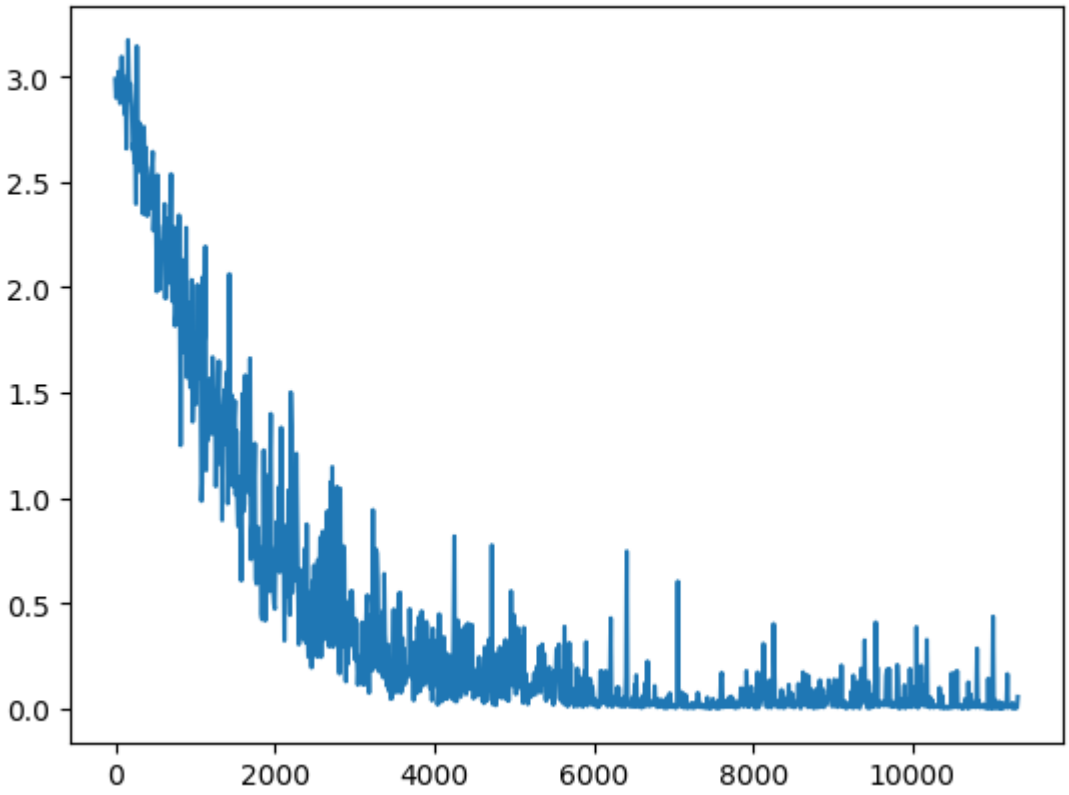
Loss	metrics	batch_size	n_epochs	device
CrossEntropyLoss	AccuracyMetric	16	20	cuda

  2、训练结果

Accuracy	Loss
0.805482	0.004088640213012695

  3、Loss曲线

  （横坐标：iteration 纵坐标：loss）



Part 2

- **关于fastNLP**

  从使用的便捷性来说，fastNLP提供的Dataset、Instance、Vocabulary模块，对于词典的构建提供了很大的便利；其次，在使用过程中，fastNLP封装的Trainer、Tester模块，对于训练、测试提供了便利，一定程度上避免了冗长的代码，同时参数更加清晰明了，便于理解和复用；模块trainer与fitlog配合使用能够便捷的保存训练过程中loss以及超参数的值。但是使用过程中发现fastNLP的文档较为欠缺，有时候需要查看源码才能清楚地了解某函数或某模块的参数以及使用方法。