PRML Assignment-4 Report

SOURCE FILES

`preprocess.py`
Codes of preprocessing the dataset.

`model.py`
Codes of TextCNN and TextRNN model.

`train.py`
Codes of training process.

QUESTION 1

PREPROCESS

- Download data from The 20 newsgroups text dataset and choose **10 categories** as the dataset of this assignment including training data and testing data.
- Use **data processing module** of FastNLP framework to preprocess the dataset, including build vocabulary dictionary in training dataset and map each word to a number.
- Split the training dataset into training dataset and development dataset.

BUILD CNN MODEL

- There are **three layers** in these model, **embedding layer**, **convolution layer** and **fully connected layer**. In the convolution layer, I use **3 different convolution kernels** respectively **3, 4, 5,** and **each** kind of convolution kernel extracts **100 features**. After **ReLU function and pooling layer**, I can get 100*3 features. Then **splice** these features and randomly **drop out** half of them. Lastly, after a fully connected layer I can get the result.

```
def __init__(self, config):
    super().__init__()
    self.embedding = nn.Embedding(config.vocab_s, config.embedding_s)
    self.convs = nn.ModuleList([
        nn.Sequential(nn.Conv2d(in_channels=1,
                                out_channels=config.feat_s,
                                kernel_size=(h, config.embedding_s)),
                      nn.ReLU(),
                      nn.MaxPool2d(kernel_size=config.max_len))
        for h in config.window_s])
    self.dropout = nn.Dropout(config.dropout_rate)
    self.fc = nn.Linear(config.feat_s*len(config.window_s), config.num_
```

BUILD RNN MODEL

- There are also **three layers** in these model, **embedding layer**, **lstm layer** and **fully connected layer**. After the lstm layer, I can get all information of hidden layer, instead of get the last one as the result, I put them into a pool layer to **get the average** of them. Then also randomly **drop out** half of them and put them into a fully connected layer to get the final result.

```
def __init__(self, config):
    super().__init__()
    self.embedding = nn.Embedding(config.vocab_s, config.embedding_s)
    self.lstm = nn.LSTM(config.embedding_s, config.hidden_s, batch_firs
    self.avg1d = nn.AvgPool1d(config.max_len)
    self.dropout = nn.Dropout(config.dropout_rate)
    self.fc = nn.Linear(config.hidden_s, config.num_class)
```

TRAIN

- Use **trainer module** and **tester module** of FastNLP framework to train and evaluate the model above. The training module will track the entire training process, and finally **load** the model which get the **best result** in development dataset in the whole epochs.

```
def train_model():
    model = Model(Config)
    loss = CrossEntropyLoss(pred="pred", target="target")
    metrics = AccuracyMetric(pred="pred", target="target")
    trainer = Trainer(model=model,
                      train_data=dataset_train,
                      dev_data=dataset_dev,
                      loss=loss,
                      metrics=metrics,
                      batch_size=16,
                      n_epochs=8)
    trainer.train()
    tester = Tester(dataset_test, model, metrics)
    tester.test()
```

- After comparing the accuracy in development dataset of several times, the **hyperparameters** of best model are as follow.

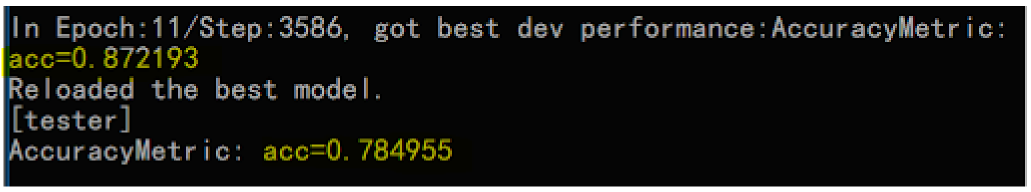
  ```
  class TextCNNConfig(object):
      num_class = class_num
      vocab_s = vocab_size
      embedding_s =  50
      feat_s =  100
      window_s = [3, 4, 5]
      max_len = max_test_len
      dropout_rate =  0.5

  class TextRNNConfig(object):
      vocab_s = vocab_size
      embedding_s =  128
      hidden_s =  128
      dropout_rate =  0.5
      num_class = class_num
      max_len = max_test_len
  ```

RESULTS

From the two pictures below, we can see the result of these two models are **very close**. But actually, the training time of RNN Model is **much longer** than CNN Model.

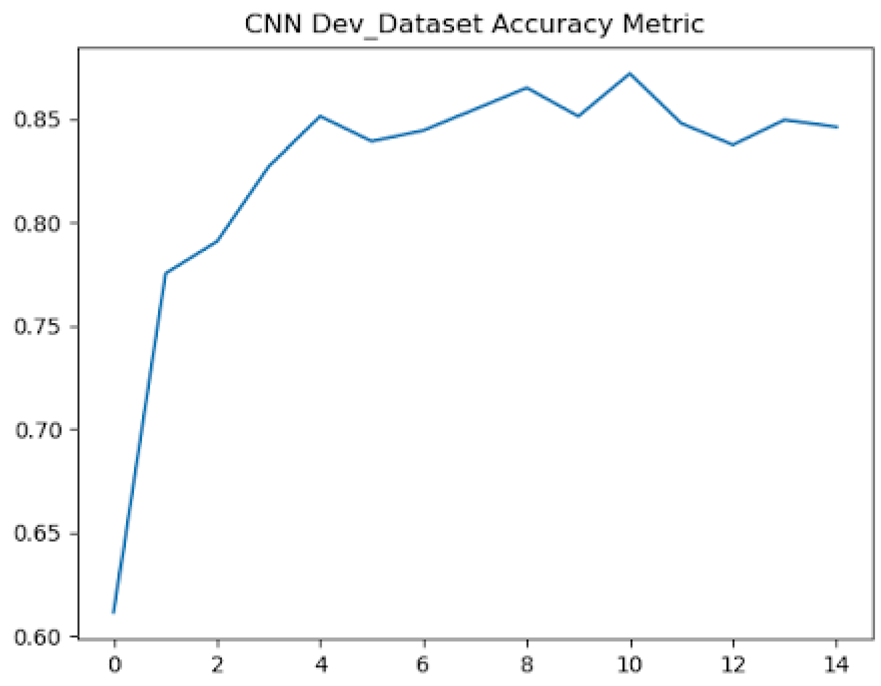
CNN Model

- In CNN Model, the best accuracy in **development dataset** is about **87.2%**, and at this time, the accuracy of **testing dataset** is about **78.5%**.



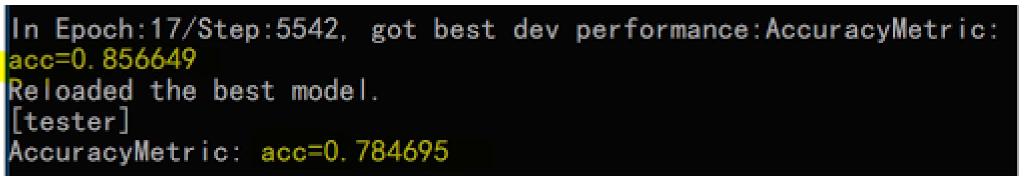
```
In Epoch:11/Step:3586, got best dev performance:AccuracyMetric:
acc=0.872193
Reloaded the best model.
[tester]
AccuracyMetric: acc=0.784955
```

- The figure below shows the **change** in accuracy of development dataset.



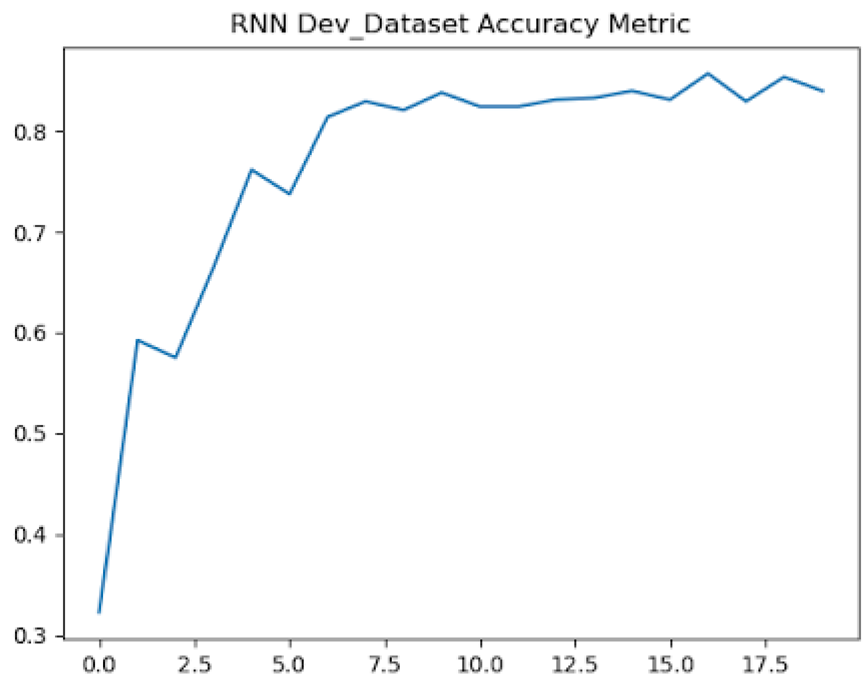
RNN Model

- In RNN Model, the best accuracy in **development dataset** is about **85.7%**, and at this time, the accuracy of **testing dataset** is about **78.5%**.



```
In Epoch:17/Step:5542, got best dev performance:AccuracyMetric:
acc=0.856649
Reloaded the best model.
[tester]
AccuracyMetric: acc=0.784695
```

- The figure below shows the **change** in accuracy of development dataset.



QUESTION 2

- After finishing the assignment 2, 3 ,4, I gradually understand the whole process of machine learning. It roughly contains three steps, preprocess the dataset, build models and train the models. In FastNLP framework, it contains processing module, training module, testing module, and some models with certain functions, which gives great convenience to us. And FastNLP is based on Pytorch, so we can use Pytorch directly. Moreover, it is easy for beginner to use it.
- As for the improvement of FastNLP, I hope the visualization of the training results can run on Windows System and the built-in models are more and more abundant.