垃圾邮件分类

数据处理与准备

与参考数据集不同，我选择的数据集是中文邮件。源数据见data/mailContent_list.pickle和mailLabel_list.pickle，共有64620封邮件，其中垃圾邮件有42854封。

考虑到GPU显存不足，故只随机选取其中1000封，保存为data/mailContent_list_1000.pickle和mailLabel_list_1000.pickle，其中垃圾邮件有659封。

随机查看一封垃圾邮件：

1. 中旅酒店预订网 http://www.zljdyd.com/ 是高科技和传统产业结合的极佳的商务网站.它性能稳定、页面清晰明了，用户预订一目了然，简单易用。本网站将以高科技的运作手段、精细化的营运模式和先进的服务理念为旅游服务企业的超常规发展拓展了新路子。网站的预订方法灵活：客户可选择上架时间、按价格、按点击率、按热门入住的方式快捷预订酒店。 作为目前中国较大的宾馆分销商，本网站费提供大中华区域内(中国大陆、香港、澳门)酒店近二百个城市 3 千多家酒店 2-7 折的超低价客房预订服务,顾客可在不输入一个字的情况下，即找到所需要的酒店。网站的目标是：利用高效的互联网技术和先进电子资讯手段，为商务客户与休闲客人提供快捷灵活、优质优惠、体贴周到又充满个性化的酒店订房服务。 常见问答： 1、预订是否收费，如何预订？ 预订服务是免费的，根据您的行程或需要，可在网上预订酒店超低价客房（2-7 折），提交您的定单后，预订中心会予您确认。所有的预订服务应尽早通知，否则可能造成预订无法顺利进行。 2、可否直接去酒店直接登记入住？ 不可以。为避免酒店客房住满等意外情况，不能未经预订而直接去酒店入住。 3、是否预订就意味着可以入住酒店了？ 不可以。必须经过预订中心的电话确认后才表示预订成功。 4、预订后如行程改变、延期、提前离店或取消预订，是否要通知预订中心？ 是的，必须通知预订中心。否则会影响您的权益。 5、预订酒店后，如何交住房款？ 入住酒店时，按 2-7 折交纳房款给入住的酒店。 中旅酒店预订网 http://www.zljdyd.com/

随机查看一封非垃圾邮件：

1. 去现场考察，路过一处坟场，荒草掩映之中石碑林立；偶尔还见到断裂的石碑，同车之人猜测大概是仇家作的事情。近年来灾害事故频繁发生，大家对鬼神的敬畏程度仍日渐淡薄，也许并不是一件好事情。 去西部调查污水排放情况。路边尽是简易的小矮房子，顺着半开的门望进去，空间极其狭小阴暗。门前的老太太背驼得很厉害，黝黑的皮肤上满是纹路；大概是因为在矿区，似乎总有洗不干净的灰尘在脸上。紧挨着的地方正在盖小高层。问她是不是回迁户，她颤巍巍地说：“以前这里的地是我们的.....”再问她将来楼房盖好了，是不是要搬进去住。她望了望，并不感兴趣的样子，说儿子们已经买了新房子，她们年纪大了，上不去楼。难道她的余生就住在这窄小的简易房中吗？这个问题没有人再去问了。 老家的农村条件也不甚好，但小孩子的记忆中自有明朗的一面，那满树鲜红的樱桃，那擦得甑亮的胆瓶.....越到大了，再看到眼下一些农村脏乱不堪的景象，每次都会受到一些刺激，竟仿若从未见过此类场景的城里人一样了，想来也觉奇怪。 晚餐中有一道“醉虾”，把活虾丢进酒和其它液体的浑浊液中，闷一会儿就可以吃了。有一次在徐州也是吃这道菜，我小心翼翼地夹了一只放在碗里，没想到这只虾酒量甚好没有醉透，拿在手中居然扭动起来，让我吃了一惊，直至终席也没敢吃上一只。这次又狭路相逢，明知道虾子还有生命，终是狠心把它放进嘴里，用牙齿直直地切下去。也没觉出什么味道来，如释重负地丢在一边。

统计得单封邮件最大字数为23945，平均字数为740，中位数为336；邮件中出现的汉字共为4427个。

处理数据集的代码文件见python/data.py中的TextData类：

（1） __init__初始化：划分训练集和测试集（7:3），定义类数、词向量维度和单封邮件最大长度（根据经验，1000个字足够判断是否为垃圾邮件）。

（2） __prepare_data获取基本信息：获取词汇表（根据出现次数倒序排列）并获得词汇量；构建词-id的映射。

（3） __content2X构建输入数据：邮件长度不足1000则在前添加padding；超过则在前删去。

（4） __label2Y构建输出数据：将label转为0和1。

（5） get_data获取数据：外部调用，返回训练输入、训练输出、测试输入、测试输出。

```
class TextData():
    # 初始化：
    # 划分训练集和测试集，定义类数、词向量维度和单封邮件最大长度
    def __init__(self, *args)

    # 获取基本信息：
    # 获取词汇表并获得词汇量；构建词-id 的映射
    def __prepare_data(self)

    # 构建输入数据：
    # 给每封邮件添加 padding
    def __content2X(self, content_list)

    # 构建输出数据：
    # 将 label 转为 0 和 1
    def __label2Y(self, label_list)

    # 获取数据：
    # 外部调用，返回训练输入、训练输出、测试输入、测试输出
    def get_data(self)
```

调用见python/main.py：

```
from data import TextData

td = TextData(content_list, label_list)
train_X, train_Y, test_X, test_Y = td.get_data()
```

模型构建

模型构建见python/cnn.py：

（1） 初始化随机词向量：将每封邮件的词id列表转化为词嵌入矩阵，(train_size, seq_length)->(train_size, seq_length, embedding_dim)。

（2） CNN卷积层：Conv1d，设定输入通道数、输出通道数和核大小，(train_size, embedding_dim, seq_length)->(train_size, num_filters, conv(seq_length))。

（3） Max Pooling最大池化层：(train_size, num_filters, conv(seq_length))->(train_size, num_filters, 1)，取切片转化为(train_size, num_filters)。

（4） 全连接层：设定隐藏层维度，(train_size, num_filters)->(train_size, hidden_dim)。

（5） Dropout层：设定保留概率。

（6） relu激活函数层。

（7） 全连接层：(train_size, hidden_dim)->(train_size, num_classes)。

```
class CNN(nn.Module):
    def __init__(self, embedding_dim, vocab_size, num_filters, kernel_size, hidden_dim, dropout_keep_prob, num_classes):
        super(CNN,self).__init__()
        self.embeds = nn.Embedding(vocab_size, embedding_dim)
        self.conv = nn.Conv1d(in_channels=embedding_dim,
                              out_channels=num_filters,
                              kernel_size=kernel_size)
        self.maxpool = nn.AdaptiveMaxPool1d(1)
        self.dense = nn.Linear(num_filters, hidden_dim)
        self.dropout = nn.Dropout(dropout_keep_prob)
        self.relu = nn.ReLU()
        self.dense2 = nn.Linear(hidden_dim, num_classes)

    def forward(self, x):
        x = self.embeds(x)
        x = x.permute(0,2,1)
        x = self.conv(x)
        x = self.maxpool(x)[:,:,0]
        x = self.dense(x)
        x = self.dropout(x)
        x = self.relu(x)
        x = self.dense2(x)
        return x
```

训练过程

模型构建见python/train.py：

（1） __init__初始化：设置随机种子，初始化模型。

（2） __cnn_build构建模型：将模型放入GPU，定义交叉熵损失函数和SGD优化器。

（3） cnn_train训练模型：外部调用，设定epoch和batch_size，保存最优模型。

（4） cnn_test测试模型：外部调用，获得测试集结果。

调用见python/main.py：

```
class TextTrain():
    # 初始化：
    # 设置随机种子，初始化模型
    def __init__(self, config, train_X, train_Y, test_X, test_Y)

    # 构建模型：
    # 将模型放入 GPU，定义交叉熵损失函数和 SGD 优化器
    def __cnn_build(self)

    # 训练模型：
    # 外部调用，保存最优模型
    def cnn_train(self)

    # 测试模型：
    # 外部调用，获得测试集结果
    def cnn_test(self)
```

考虑GPU显存容量，设定batch_size为100，epoch为400。

```
config = {
    # train
    'num_iteration': 300,
    'batch_size': 100,
    'learning_rate': 1e-3,
    'print_per_batch': 500 / 10,
    # model
    'embedding_dim': 64,
    'num_filters': 256,
    'kernel_size': 5,
    'hidden_dim': 128,
    'dropout_keep_prob': 0.5
}

train = TextTrain(config, train_X, train_Y, test_X, test_Y)
train.cnn_train()
```

训练结果为：

```
step:1 loss:0.7124 accuracy:0.5100
step:50 loss:0.6199 accuracy:0.6500
step:100 loss:0.5470 accuracy:0.6900
step:150 loss:0.4700 accuracy:0.8000
step:200 loss:0.4064 accuracy:0.7900
step:250 loss:0.2650 accuracy:0.9300
step:300 loss:0.1761 accuracy:0.9400
step:350 loss:0.1028 accuracy:0.9800
step:400 loss:0.1189 accuracy:0.9800
```

测试集结果

测试集结果见python/train.py：

（1） evaluate：计算loss和accuracy。

（2） print_report_table：计算Precision、Recall、F1和Support。

（3） print_confusion_matrix：计算混淆矩阵。

（4） plot_roc_curve：绘制ROC曲线。

```
class TextTrain():
    # 计算 loss 和 accuracy
    @staticmethod
    def evaluate(criterion, o, y)

    # 计算 Precision、Recall、F1 和 Support
    @staticmethod
    def print_report_table(o, y, labels)

    # 计算混淆矩阵
    @staticmethod
    def print_confusion_matrix(o, y, labels)

    # 绘制 ROC 曲线
    @staticmethod
    def plot_roc_curve(o, y)
```

调用见python/main.py：

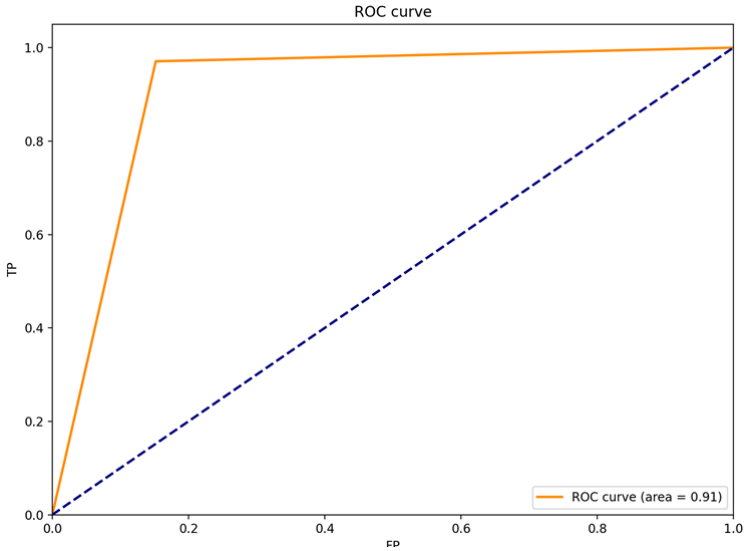
```
train.cnn_test()
```

测试结果为：

```
test loss:0.2274 accuracy:0.9320

 Report Table:
   Label  Precision    Recall        F1  Support
0   ham   0.930556  0.848101  0.887417       79
1  spam   0.932584  0.970760  0.951289      171
2    总体   0.931943  0.932000  0.931106      250

 Confusion Matrix:
       ham  spam
ham    67    12
spam    5   166
```


ROC curve
TP
FP
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.91)