**Pytorch数据加载-文本数据加载**

在上次的文章中我们讲述了Python数据加载高维数据的加载，主要是加载图片数据集，详情可以从以下文章中进行查看：

**Pytorch数据加载-图像数据的加载**

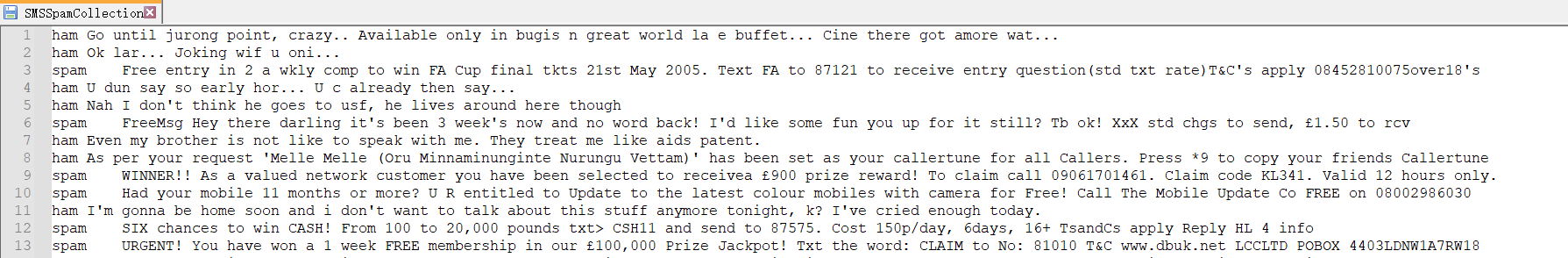
在实际的工程项目中，我们需要加载文本数据进行任务的训练，比如我们做一个情感分析的项目的时候，毕竟文本数据也是我们非常常见的数据形式了。

1. **txt文本数据加载-一般方法**

这里我们使用的文本数据是一个垃圾短信数据集SMSSpamCollection，大家可以从这个链接中进行下载：

<http://archive.ics.uci.edu/ml/machine-learning-databases>

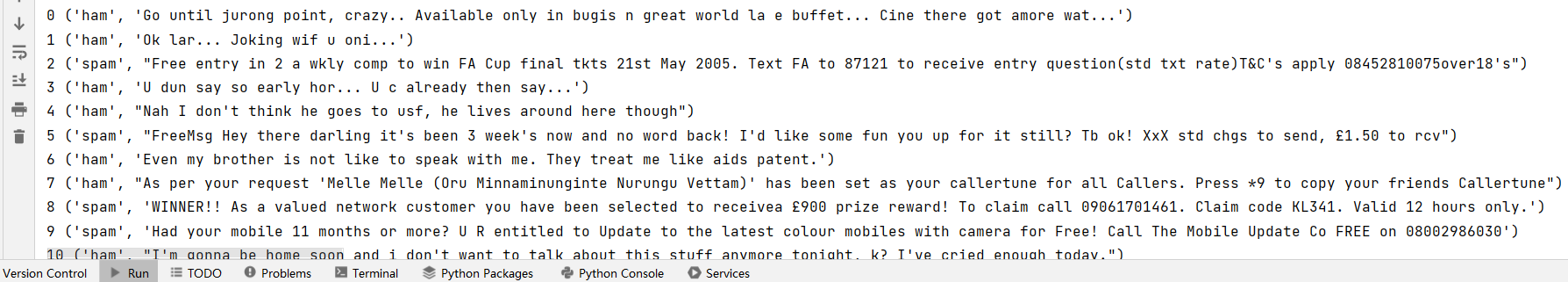
SMSSpamCollection分类任务数据集，是用于骚扰短信识别的经典数据集，文本数据是基于真实的短信内容，有5574条数据，包括正常短信和骚扰短信数据，每行完整记录一条短信内容，每行开头通过ham和spam标识正常短信和骚扰短信，如下所示：



接下来我们编写代码进行数据读取：

from torch.utils.data import Dataset,DataLoader  
import pandas as pd  
  
class MyDataset(Dataset):  
 def \_\_init\_\_(self,data\_path):  
 self.data\_path = data\_path  
 self.df = pd.read\_csv(self.data\_path, sep='\t', header=None)  
 new\_col = ['label', 'sms']  
 self.df.columns = new\_col  
  
 def \_\_getitem\_\_(self, index):  
 item = self.df.iloc[index,:]  
 return item.values[0],item.values[1]  
  
 def \_\_len\_\_(self):  
 return self.df.shape[0]  
  
  
data\_path = r"SMSSpamCollection"  
d = CifarDataset(data\_path)  
for idx in range(len(d)):  
 *# print(idx,d.\_\_getitem\_\_(idx))* print(idx, d[idx])

这里使用循环打印出了这个数据集下所有行的数据，输出如下所示：



注意在循环中：

print(idx,d.\_\_getitem\_\_(idx))  
print(idx, d[idx])

这两行代码的效果是一样的，都可以输出特定行的数据，还是比较方便的。这样我们就完成了文本数据的加载，当然了文本数据加载之后还要进行一定的清洗和向量转换才能被模型所接受。

使用上述方法我们可以完成数据集的加载，当是存在一个问题就是无法进行批量架加载数据集和在加载数据集的时候同时进行打乱的操作。

1. **txt文本数据加载-DataLoader**

接下来我们借助torch.utils.data.DataLoader来进行数据的加载，直接看一下代码然后在做解释。

from torch.utils.data import Dataset,DataLoader  
import pandas as pd  
  
class MyDataset(Dataset):  
 def \_\_init\_\_(self,data\_path):  
 self.data\_path = data\_path  
 self.df = pd.read\_csv(self.data\_path, sep='\t', header=None)  
 new\_col = ['label', 'sms']  
 self.df.columns = new\_col  
  
 def \_\_getitem\_\_(self, index):  
 item = self.df.iloc[index,:]  
 return item.values[0],item.values[1]  
  
 def \_\_len\_\_(self):  
 return self.df.shape[0]  
  
data\_path = r"SMSSpamCollection"  
mydataset = MyDataset(data\_path)  
data\_loader = DataLoader(dataset=mydataset, batch\_size=20, shuffle=True)  
print("总数据长度:",len(mydataset))  
for idx,(label,sms) in enumerate(data\_loader):  
 if idx < 1:  
 print("标签:",label)  
 print("数据:",sms)  
 print("本次批数据对象:",data\_loader)  
 print("本次批数据长度:",len(data\_loader))  
 print('\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_')

输出如下：



从代码角度上来看，使用dataloader加载文本数据集与加载图像数据集是类似的，这里也稍微解释一下代码：

1. 在data\_loader中我们使用参数batch\_size=20，所以每批次的数量为5572/20 = 278.6

进行四舍五入一下就是278。

1. batch\_size=20，所以标签和数据的输出都是20个。

另外推荐一个torchtext模块，小伙伴们在使用之前记得install。稍微介绍一下这个torchtext模块，其主要的作用就是：

**将[NLP](https://so.csdn.net/so/search?q=NLP&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/weixin_39573512/article/details/_blank)中从文本转换为Tensor的整个流程进行了封装**

一般来说有以下的一些作用：

（1）进行数据的预处理

（2）分批次，填充，及数值化(Tensor转换)

（3）训练、验证、测试数据集的划分

具体的一些方法和使用大家可以参考官方文档进行学习，在后面的教程中如果有使用到也会进行补充。

1. **参考文档**

<https://pytorch.org/docs/stable/data.html>

https://mp.weixin.qq.com/s?\_\_biz=MzU1MzEzOTc4Nw==&amp;mid=2247496919&amp;idx=1&amp;sn=3f6124b8c4c37d2df9f3363b4088b477&amp;chksm=fbf5dff3cc8256e5be9d6b2249e75590c2328f567e2e467437ef295a9afc005ecb71cdc7c06f&token=1435288653&lang=zh\_CN#rd