对于不同的数据集我们要使用不同的processor，对于使用csv文件的我们使用Myprocessor如果有自己的数据格式我们可以自己定义一个processor类继承基类，并且修改其中几个参数

def get\_labels(self):  
 *"""See base class."""* return ["0", "1"]

上面的方法中的标签类型应该根据不同数据进行修改

def \_create\_examples(self, lines, set\_type):  
 *"""Creates examples for the training and dev sets."""* examples = []  
 for (i, line) in enumerate(lines):  
 # Only the test set has a header  
 if i == 0:  
 continue  
 guid = "%s-%s" % (set\_type, i)  
  
 text\_a = tokenization.convert\_to\_unicode(line[0])  
 label = tokenization.convert\_to\_unicode(line[1])  
 examples.append(  
 InputExample(guid=guid, text\_a=text\_a, text\_b=None, label=label))  
 return examples

上述标红字段中text\_a表示的是文本段其中文本在哪一列就要改到那一列

标红中的label表示的是数据的标签要改到标签所在行

def \_read\_tsv(cls, input\_file, quotechar=None):  
 *"""Reads a tab separated value file."""* with tf.gfile.Open(input\_file, "r") as f:  
 reader = csv.reader(f, delimiter=",", quotechar='"') # quotechar  
 lines = []  
 for line in reader:  
 lines.append(line)  
 return lines

要注意的是在上述代码的quotechar这个参数代表的是转义字符，例如我们的分隔符delimiter是逗号，如果我们处理的文本中有逗号，那么就需要转义字符将其括起来（上述方法在Dataprocessor基类中）

下述几个参数：

Required parameters是必须要设置的参数不能为none

## Required parameters  
flags.DEFINE\_string(  
 "data\_dir", "C:/Users/dell/Desktop/学姐数据/2\_dimen",  
 "The input data dir. Should contain the .tsv files (or other data files) "  
 "for the task.")  
  
flags.DEFINE\_string(  
 "bert\_config\_file","C:/Users/dell/Desktop/bert-master/pretrain\_data/bert\_config.json",  
 "The config json file corresponding to the pre-trained BERT model. "  
 "This specifies the model architecture.")  
  
flags.DEFINE\_string("task\_name", "mine", "The name of the task to train.")  
  
flags.DEFINE\_string("vocab\_file", "C:/Users/dell/Desktop/bert-master/pretrain\_data/vocab.txt",  
 "The vocabulary file that the BERT model was trained on.")  
  
flags.DEFINE\_string(  
 "output\_dir", "C:/Users/dell/Desktop/学姐数据/2\_dimen\_result",  
 "The output directory where the model checkpoints will be written.")

data\_dir是数据文件所在的文件夹（文件夹中必须要包括train文件用来训练dev文件用来预测和test文件用来评估）

bert\_config\_file参数需要设置为你预先下载的已经预训练好的模型的参数配置所在的文件夹

task\_name你所进行的任务类型（我的任务是进行多分类使用的是csv文件所以用mine）

vocab\_file也是预训练模型文件之一（没搞明白干啥的）

output\_dir输出文件夹在的位置

然后Other parameters是可选的参数

flags.DEFINE\_string(  
 "init\_checkpoint", None,  
 "Initial checkpoint (usually from a pre-trained BERT model).")  
  
flags.DEFINE\_bool(  
 "do\_lower\_case", True,  
 "Whether to lower case the input text. Should be True for uncased "  
 "models and False for cased models.")  
  
flags.DEFINE\_integer(  
 "max\_seq\_length", 26,  
 "The maximum total input sequence length after WordPiece tokenization. "  
 "Sequences longer than this will be truncated, and sequences shorter "  
 "than this will be padded.")  
  
flags.DEFINE\_bool("do\_train", True, "Whether to run training.")  
  
flags.DEFINE\_bool("do\_eval", True, "Whether to run eval on the dev set.")  
  
flags.DEFINE\_bool(  
 "do\_predict", True,  
 "Whether to run the model in inference mode on the test set.")  
  
flags.DEFINE\_integer("train\_batch\_size", 32, "Total batch size for training.")  
  
flags.DEFINE\_integer("eval\_batch\_size", 8, "Total batch size for eval.")  
  
flags.DEFINE\_integer("predict\_batch\_size", 8, "Total batch size for predict.")  
  
flags.DEFINE\_float("learning\_rate", 2e-5, "The initial learning rate for Adam.")  
  
flags.DEFINE\_float("num\_train\_epochs", 3.0,  
 "Total number of training epochs to perform.")

很多只挑几个比较重要的讲

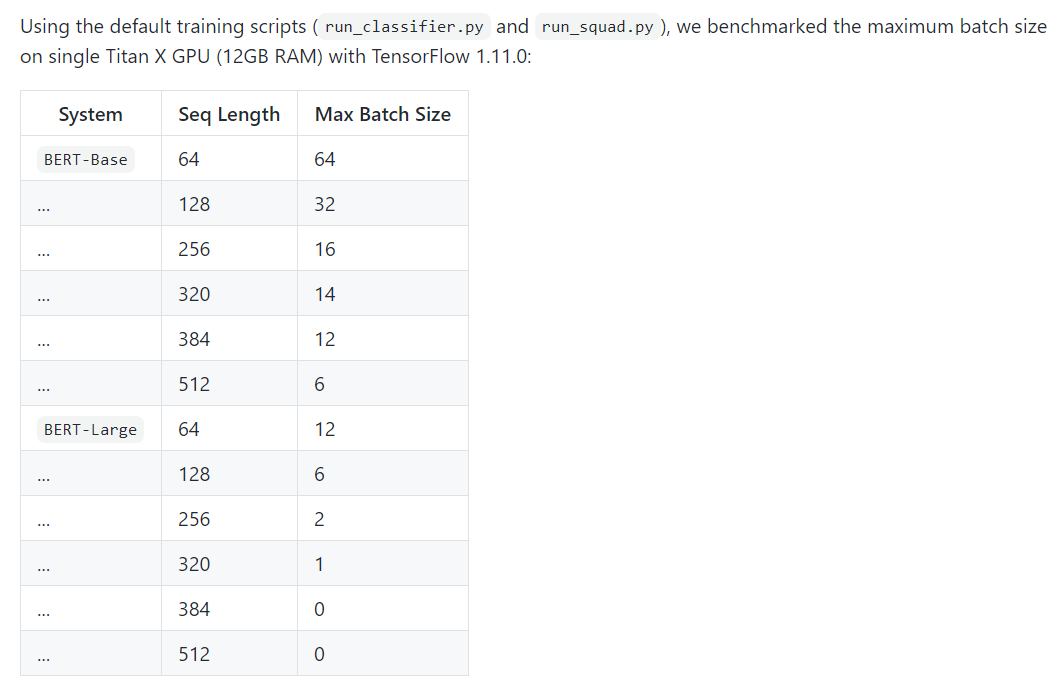
max\_seq\_length就是你设置的句子的最大长度，如果字数超过这个长度就会被截断

do\_train do\_eval do\_predict这三个参数就是训练，评估，和预测至少有一个为Ture

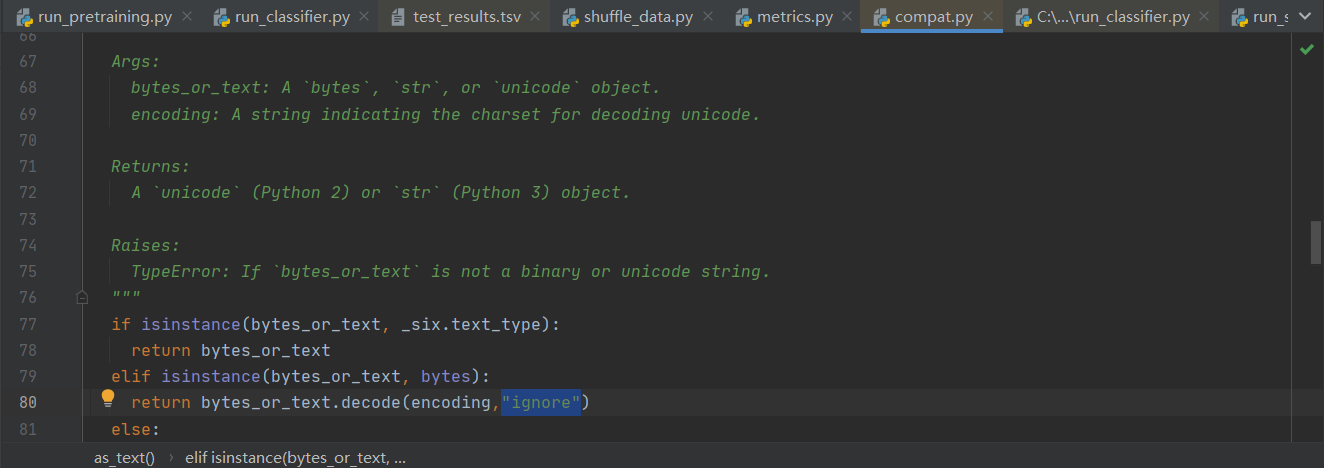
learning\_rate学习率值越低，沿着梯度下降越慢。花费很长的时间来收敛。学习率会影响我们的模型能够以多快的速度收敛到局部最小值（也就是达到最好的精度）。 因此，从正确的方向做出正确的选择意味着我们能用更少的时间来训练模型。

batch\_size我们用来更新梯度的批数据大小。一般来说，Batch Size设置的不能太大也不能太小，一般为几十或者几百。

epoch：1个epoch指用训练集中的全部样本训练一次，此时相当于batchsize 等于训练集的样本数



遇到报错可以尝试修改这个文件中的高亮字段



<https://blog.csdn.net/zyq12345678/article/details/89419752>

shuffle\_size也需要根据数据集进行更改

如果训练样本顺序输入，且同类样本较多时，代码中的shuffle可能起不到作用。d = d.shuffle(buffer\_size=100)使用了Dataset的shuffle，其中buffer\_size指缓冲区大小。训练时每次从缓冲区中按batch取样本，但缓冲区的补充是顺序取后续样本，如果buffer\_size=1那就是没有打乱，顺序地取样本，如果buffer\_size等于样本总数就相当于全局shuflle，但可能内存不够。

【建议】：数据处理时提前打乱；或buffer\_size调大些，比如调成最大类下样本数。

def input\_fn(params):  
 *"""The actual input function."""* batch\_size = params["batch\_size"]  
  
 # For training, we want a lot of parallel reading and shuffling.  
 # For eval, we want no shuffling and parallel reading doesn't matter.  
 d = tf.data.TFRecordDataset(input\_file)  
 if is\_training:  
 d = d.repeat()  
 d = d.shuffle(buffer\_size=2895)  
  
 d = d.apply(  
 tf.contrib.data.map\_and\_batch(  
 lambda record: \_decode\_record(record, name\_to\_features),  
 batch\_size=batch\_size,  
 drop\_remainder=drop\_remainder))