**multi\_Alstm\_classify项目详细说明书**

**一、总述**

run\_att\_bilstm.sh是运行训练脚本

run\_predict.sh是运行预测脚本

models/attn\_bi\_lstm.py是训练文件

models/predict.py是预测文件

models/Config/config.cfg是配置文件

models/utils/data\_helper.py是数据加载文件

models/utils/model\_helper.py是执行函数文件

ckpt2pb.py是转pb模型文件

执行需要修改的相关信息在config.cfg文件中，log文件在当日日期文件夹下，训练和预测需要更改config中相关参数

训练或预测必须要修改config.cfg中的以下信息：

data\_dir：数据存放路径 训练验证测试数据文件分别是：${cls\_type}\_train.csv、${cls\_type}\_test.csv、${cls\_type}\_predict.csv 其中cls\_type为具体类别，for example：ask\_know\_train.csv、ask\_know\_test.csv、ask\_know\_predict.csv 每个文件需要有use\_cols=['type\_robot','msg','type','type\_combine']列，for example：['询问是否认识借款人','我认识','肯定','yes'] save\_dir：模型保存路径is\_train = True ：True为训练，False为预测path\_model\_Predict：预测模型的路径（保存模型的路径）

run\_att\_bilstm.sh 运行训练脚本中的python自行修改 run\_predict.sh 运行预测脚本中的python自行修改

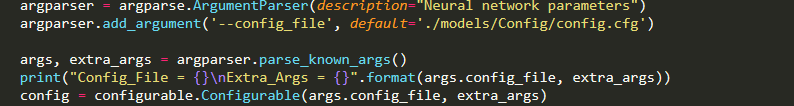
ckpt2pb.py 转pb模型文件 需要修改模型路径，根据训练模型中的类别（cls\_type）信息修改需要预测的out\_put

1. **models/attn\_bi\_lstm.py训练文件相关函数说明**

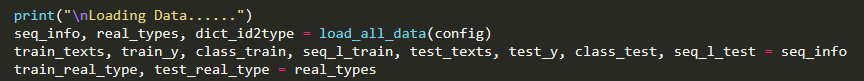
***if \_\_name\_\_ == '\_\_main\_\_': 主函数***

功能如下：

1. 通过路径加载config配置文件



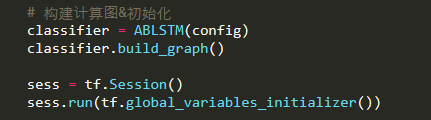
1. 通过读取配置文件加载数据集（通过load\_all\_data函数）



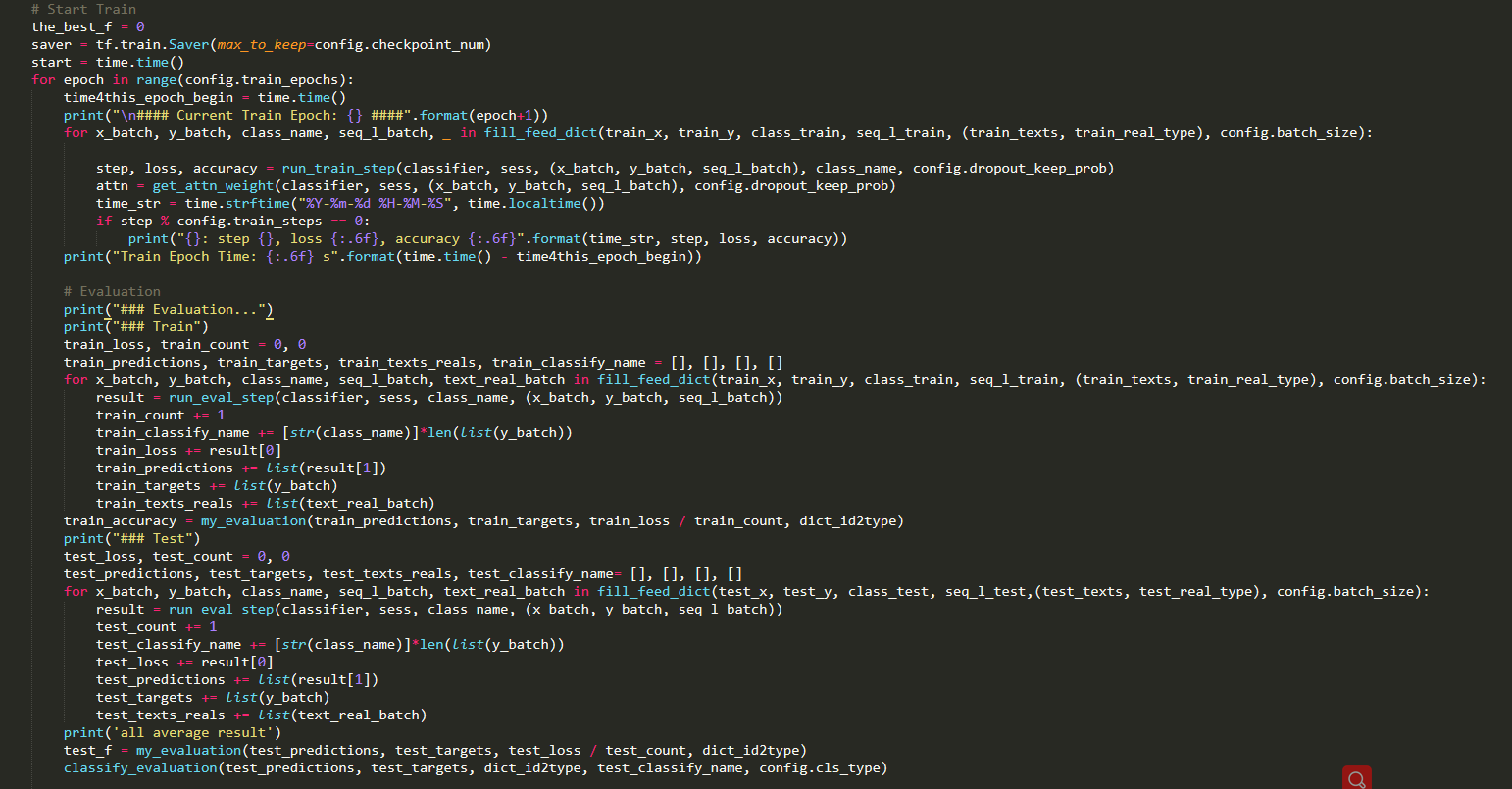
1. 对数据进行to\_sequences处理（通过data\_preprocessing\_v2函数）



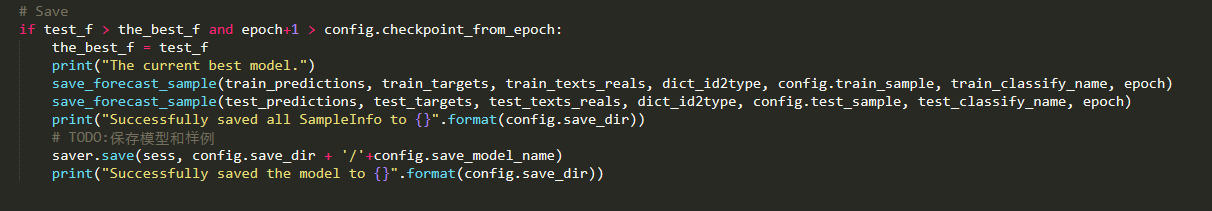
1. 构建计算图&初始化（通过ABLSTM类build\_graph函数）



1. 开始训练和评估操作



1. 保存模型



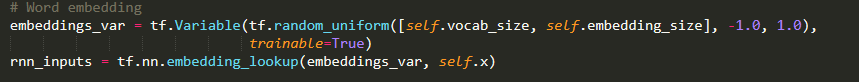
***class ABLSTM(object):*** ***BiLstm算法模型类***

***def \_\_init\_\_(self, config):***

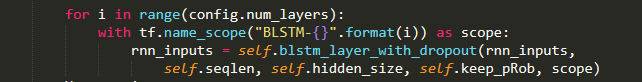
1. 根据配置文件初始化参数，定义模型输入

***def build\_graph(self): 构建网络：***

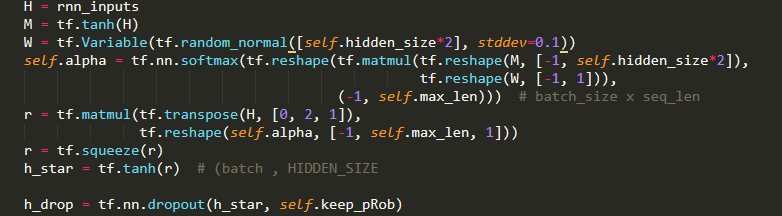
1. 搭建embedding层



1. 搭建BiLstm算法结构（blstm\_layer\_with\_dropout函数）



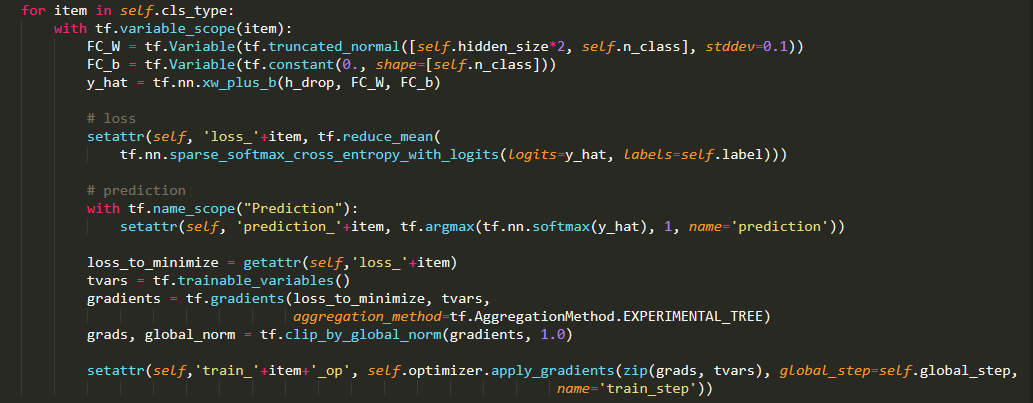
1. 搭建激活函数、attention、Dropout操作



1. 定义优化器

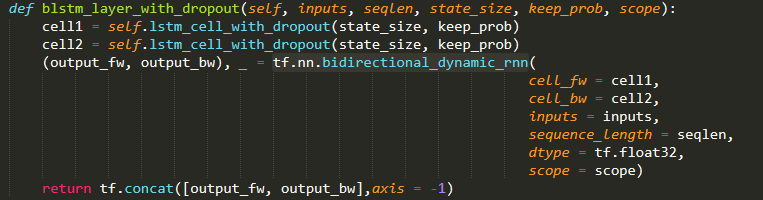


1. 根据多个任务，分别搭建优化函数



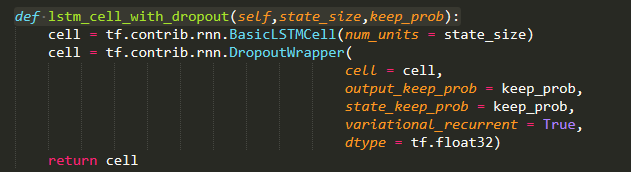
***def blstm\_layer\_with\_dropout(self, inputs, seqlen, state\_size, keep\_prob, scope): BiLstm 网络搭建***

调用lstm\_cell\_with\_dropout函数创建前向lstm和后向lstm，通过tf.nn.bidirectional\_dynamic\_rnn 创建bilstm。

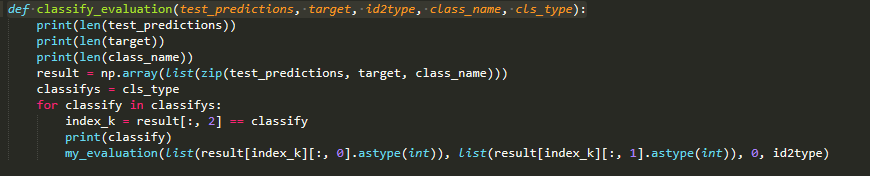


***def lstm\_cell\_with\_dropout(self,state\_size,keep\_prob): Lstm+dropout***

定义lstm+Dropout



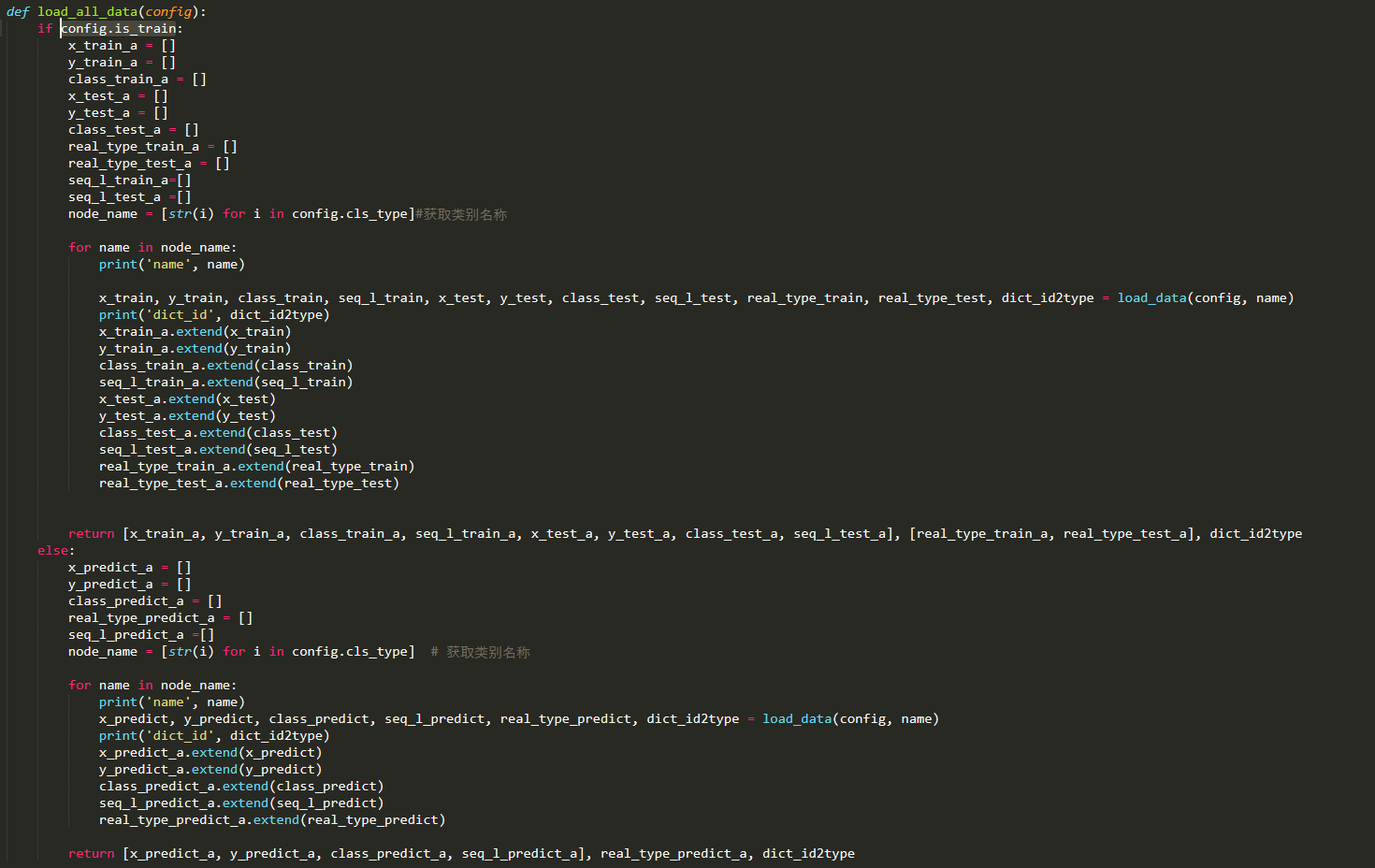
***def classify\_evaluation(test\_predictions, target, id2type, class\_name, cls\_type):对每个任务进行评估。调用my\_evaluation函数***



1. **models/utils/data\_helper.py是数据加载文件相关函数说明**

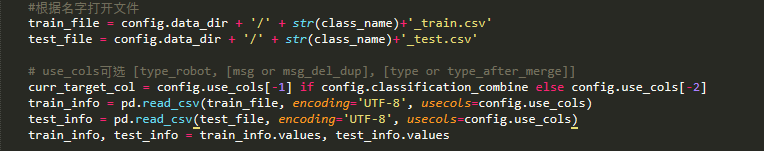
***def load\_all\_data(config):数据加载主函数***

该函数通过类别名调用***load\_data***进行数据加载，然后将返回的数据添加到list中，分为训练数据加载和预测数据加载，通过配置文件中config.is\_train来控制，主要区别为，训练需要加载训练和验证两个数据集，预测评估只要加载预测数据集。

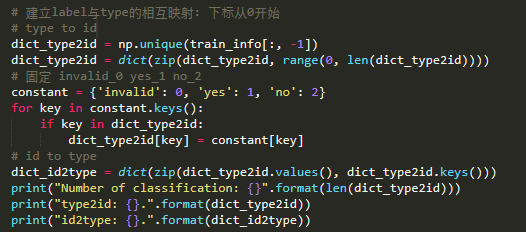


***def*** ***load\_data(config, class\_name):根据类别加载数据文件***

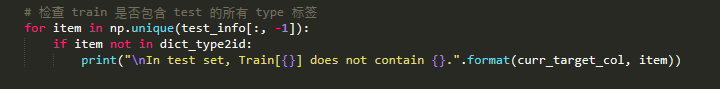
1. 打开文件，加载指定列信息



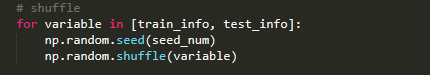
1. 建立label与type（文字lable）的相互映射



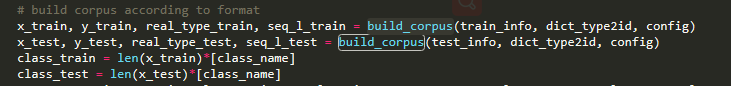
1. 检查 train 是否包含 test 的所有 type 标签



1. shuffle操作

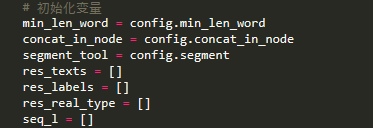


1. 数据分词，调用build\_corpus函数

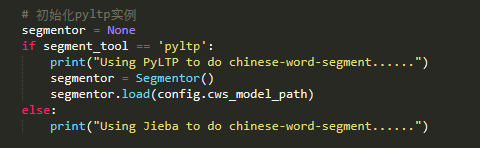


***def build\_corpus(seq\_info, dict\_type2label, config):分词处理***

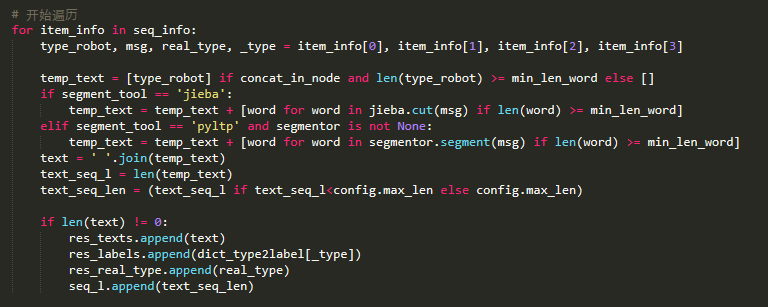
1. 通过配置文件初始化相关参数



1. 初始化pyltp实例

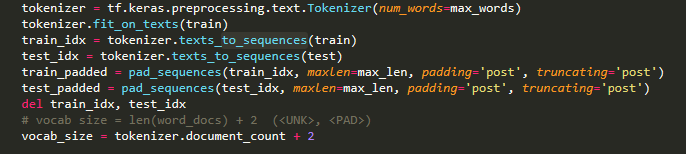


1. 开始遍历分词，并记录有效长度

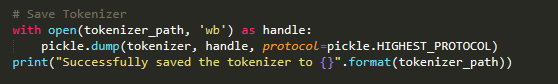


***def data\_preprocessing\_v2(train, test, tokenizer\_path, max\_len, max\_words=50000):，建立tokenizer，处理数据，保存tokenizer***

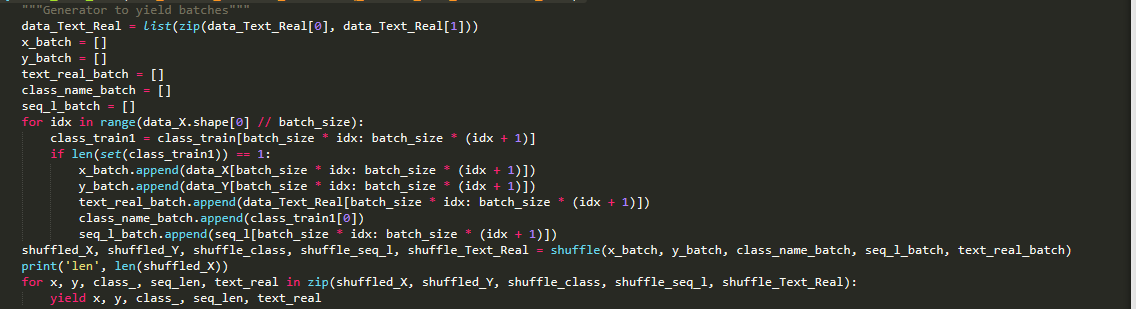
1. 通过数据建立Tokenizer，处理数据to\_sequences，增加padding操作



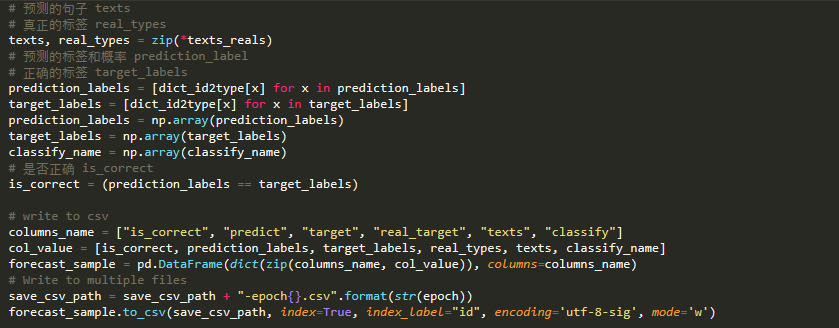
1. 保存Tokenizer（预测评估需要使用）



***def build\_corpus(seq\_info, dict\_type2label, config):产生batch数据，并打乱数据。***

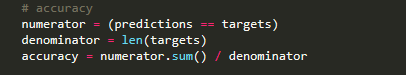


***def save\_forecast\_sample(prediction\_labels, target\_labels, texts\_reals, dict\_id2type, save\_csv\_path, classify\_name, epoch):保存算法预测样例数据***

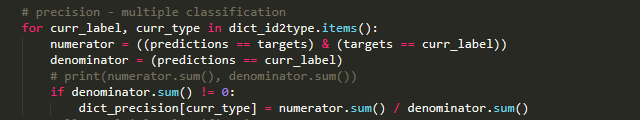


***def my\_evaluation(predictions, targets, loss, dict\_id2type):评估函数***

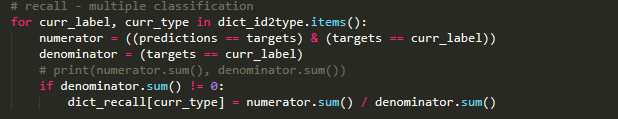
1. 计算accuracy



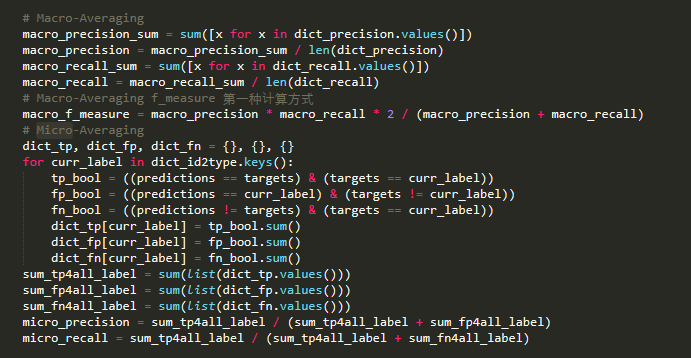
1. 按照分类计算precision



1. 按照分类计算recall

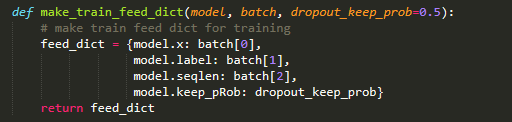


1. 求Averaging，（两种方式）

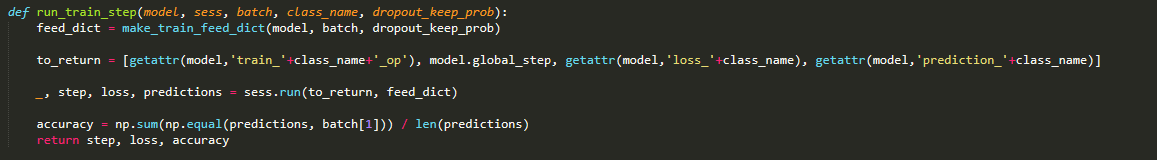


1. **models/utils/model\_helper.py是数据加载文件相关函数说明**

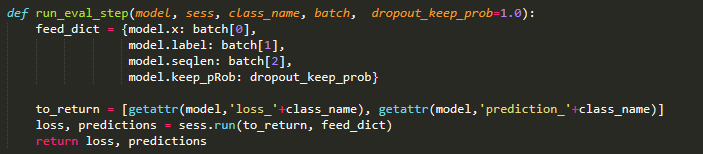
***def make\_train\_feed\_dict(model, batch, dropout\_keep\_prob=0.5):给算法模型传入数据***



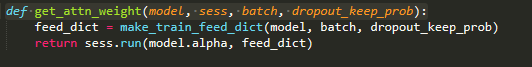
***def run\_train\_step(model, sess, batch, class\_name, dropout\_keep\_prob):训练传入数据，对每个任务进行优化，返回步长、loss、accuracy***



***def run\_eval\_step(model, sess, class\_name, batch, dropout\_keep\_prob=1.0)验证传入数据，返回loss、预测结果***



***def get\_attn\_weight(model, sess, batch, dropout\_keep\_prob): Attention权重***

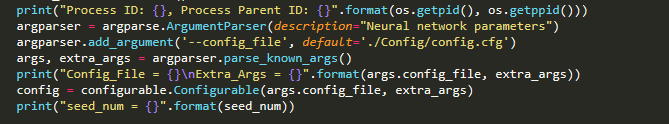


1. **models/predict.py预测文件相关函数说明**

***if \_\_name\_\_ == '\_\_main\_\_': 主函数***

功能如下：

1、通过路径加载config配置文件



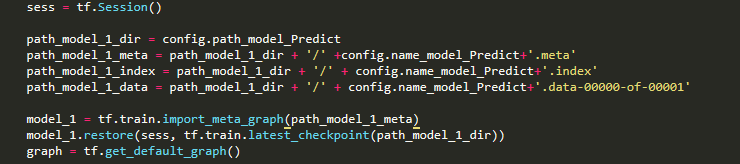
2、通过读取配置文件加载数据集（通过load\_all\_data函数）



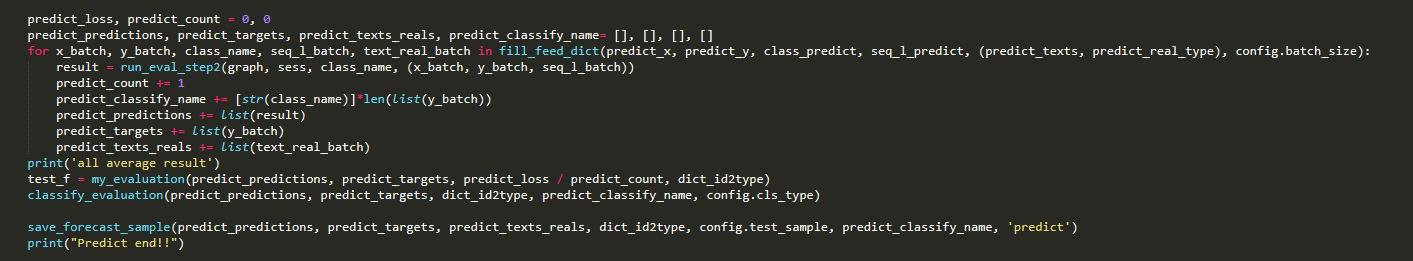
3、对数据进行to\_sequences处理（通过data\_preprocessing\_v2函数）



4、加载模型

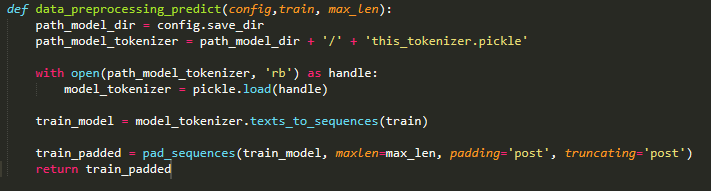


5、预测评估

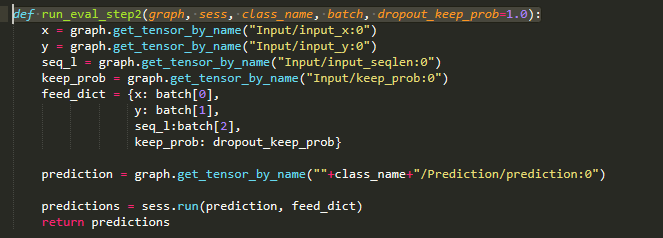


***def data\_preprocessing\_predict(config,train, max\_len):预测数据处理***

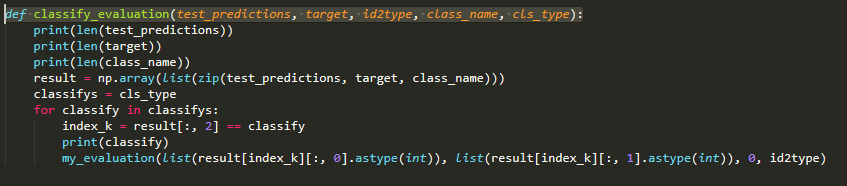
***，加载Tokenizer做to\_sequences，增加padding***



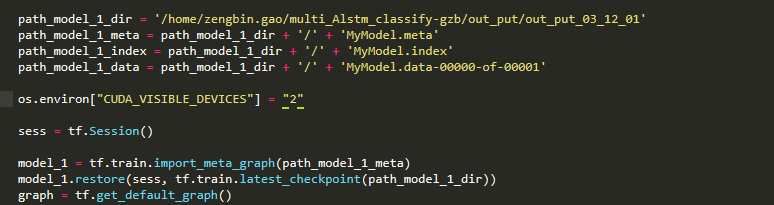
***def run\_eval\_step2(graph, sess, class\_name, batch, dropout\_keep\_prob=1.0):对模型传入数据，获取预测结果。***

******

***def classify\_evaluation(test\_predictions, target, id2type, class\_name, cls\_type):按任务计算相关评估参数，调用my\_evaluation函数***



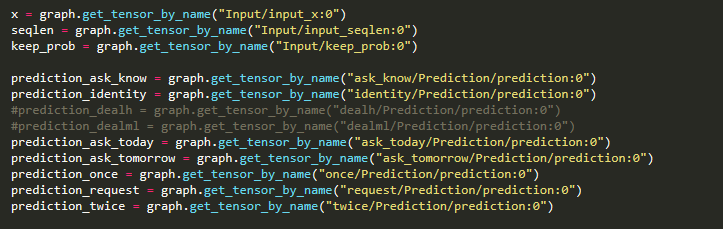
1. **ckpt2pb.py 转pb模型文件说明**
2. 加载ckpt模型，加载图



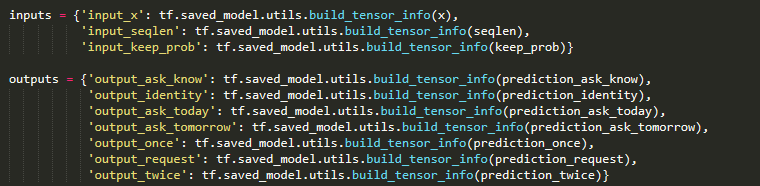
1. 设置pb模型保存路径



1. 获取模型的输入和输出（任务数量决定）



1. 编写pb模型的输入输出



1. 保存pb模型

