目标：输入LSTM的一个batch的数据可以是不同长度的，有过操作是不同长度的句子用0来padding，缺点是。。

方法：

Pytorch提供了两个函数：

pad\_sequence(传入一个list)自动可以做到

torch.nn.utils.rnn.pack\_padded\_sequence

(x,input\_lengths, batch\_first=True,enforce\_sorted=False)

X是一个batch的数据，里面是pad\_sequence

Lengths是一个list，按顺序存储了每个数据的长度

Batch\_fist告诉它是不是第一维度是batch\_size

Enforced\_sorted告诉它你的x是不是按照长度降序排列的，不是的话传false它会自己排

具体应用场合：

首先：

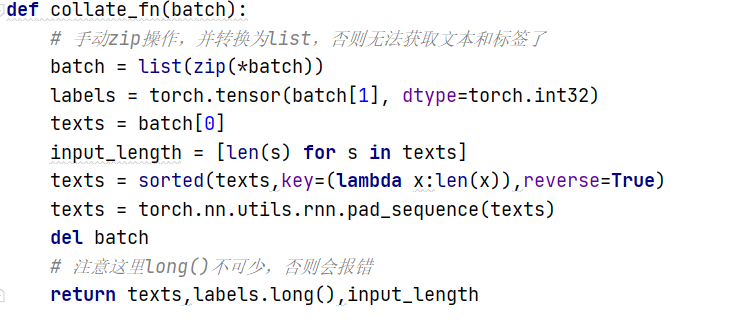
train\_loader = DataLoader(dataset=train\_dataset, batch\_size=64, shuffle=**True**,collate\_fn=collate\_fn)

这一步是可以在batch里面都是不同长度的tensor的时候成功运行的

也就是说batch的形成是可以比如64个样本都是不同长度的，但是取不出来

我们要做的是自定义 collate\_fn函数，规定怎么取

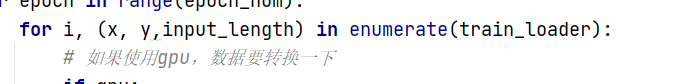
取的时候



回复值包含一个input\_length数组，同时要将tensor包裹成pad\_sequence

最好将texts和input\_length都用降序排好，便于后面pack\_padded\_sequence好用。

这样在用data\_loader时

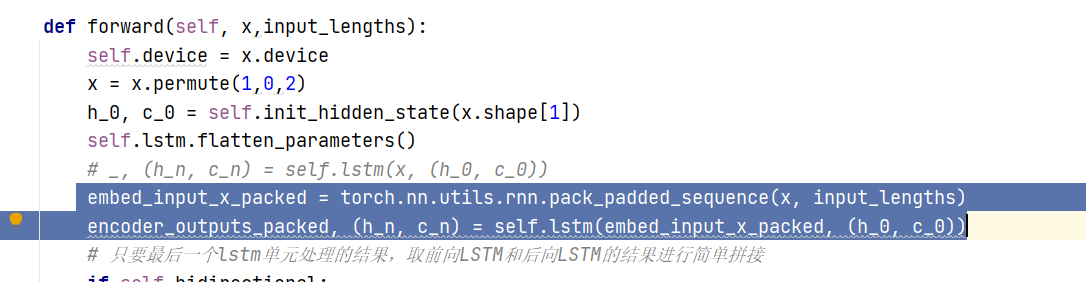


同样 input\_length要传给模型的forward

方法时：

直接这样写就行了，forward也这样定义

然后在forward里面：



可以看到，就可以用pack\_padded\_sequence了，结果可以直接输入给LSTM

尺寸问题：

如果要多卡gpu训练，则输入的x的 第一维必须是batch\_size，这是正常dataloader的设定，

但是如果你自定义了一个coll\_fn，则一定要检查一下，因为这里面的

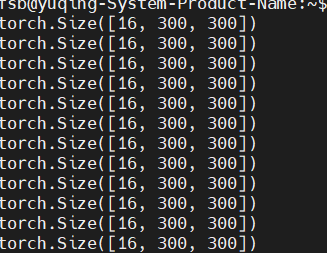


输出的batch\_size在第二维，

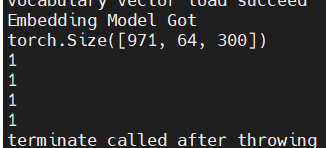
所以，我们需要手动将其调到第一维



正常的ddp的输入到了forward里面的shape：（自动分了4块卡，shape[0]=64/4）



而对于自己写的，forward前的shape：

batch\_size在第二维，这样无法多卡

注：input\_length也要调整成cuda类型