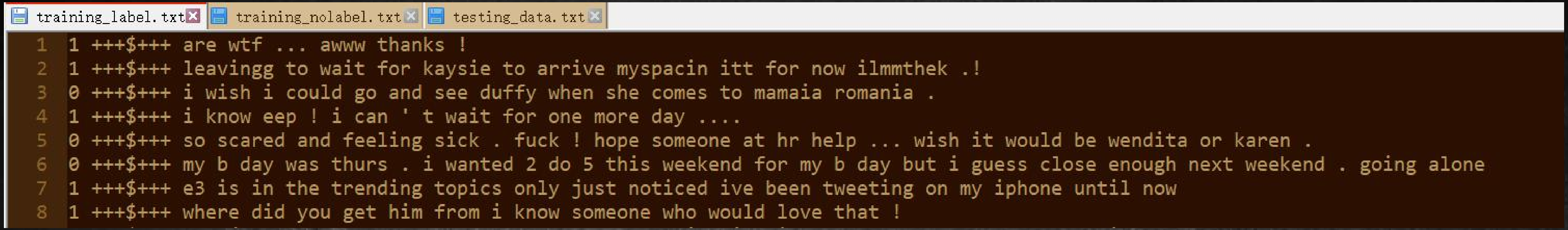
学习汇报

焦强

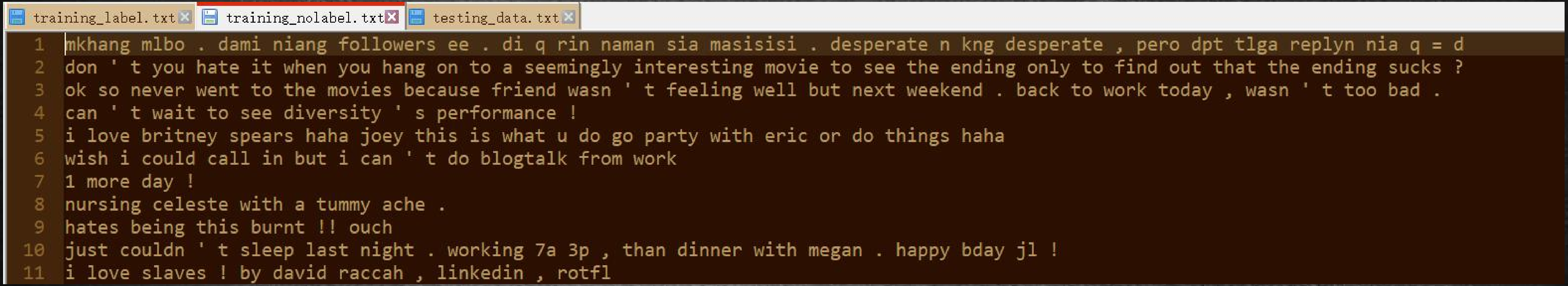
文本分类

通过循环神经网络对句子进行情感分类。

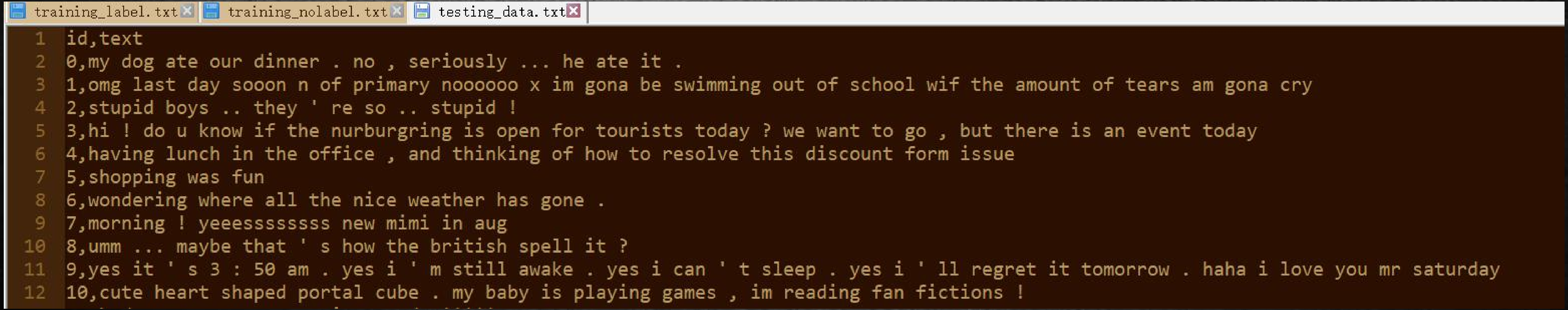
给定一句子，判断这句子是正面还是负面的



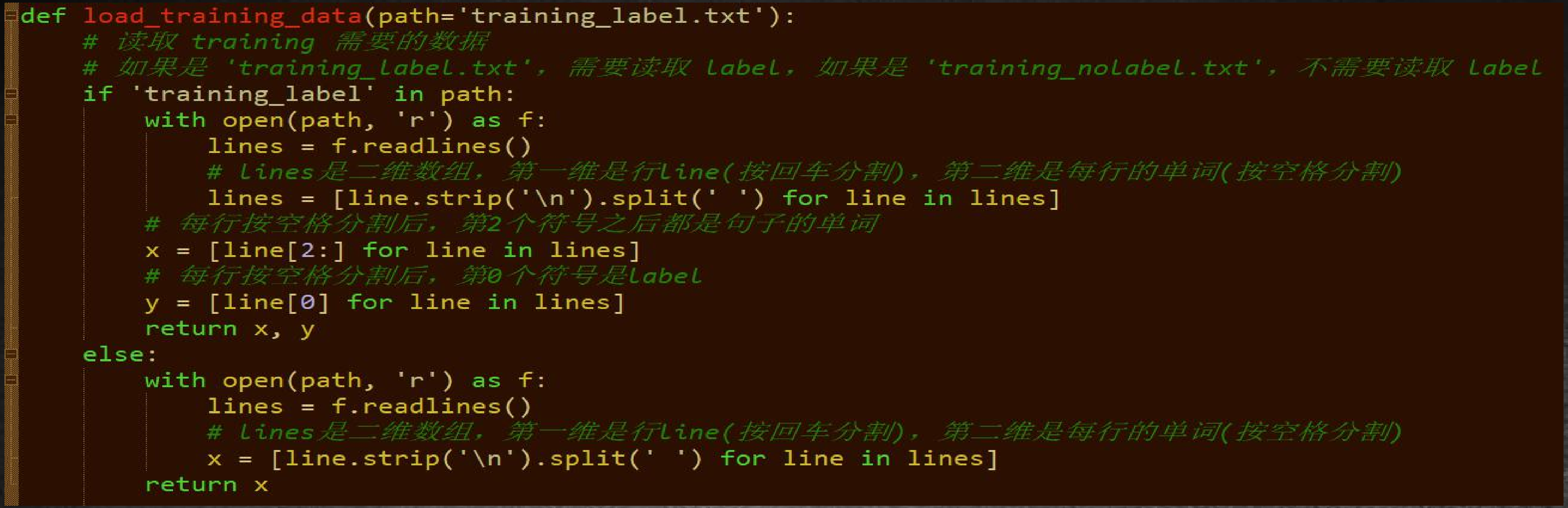
```
1 +++$+++ are wtf ... awww thanks !
1 +++$+++ leavingg to wait for kaysie to arrive myspacin itt for now ilmmthek .!
0 +++$+++ i wish i could go and see duffy when she comes to mamaia romania .
1 +++$+++ i know eep ! i can ' t wait for one more day ....
0 +++$+++ so scared and feeling sick . fuck ! hope someone at hr help ... wish it would be wendita or karen .
0 +++$+++ my b day was thurs . i wanted 2 do 5 this weekend for my b day but i guess close enough next weekend . going alone
1 +++$+++ e3 is in the trending topics only just noticed ive been tweeting on my iphone until now
1 +++$+++ where did you get him from i know someone who would love that !
```



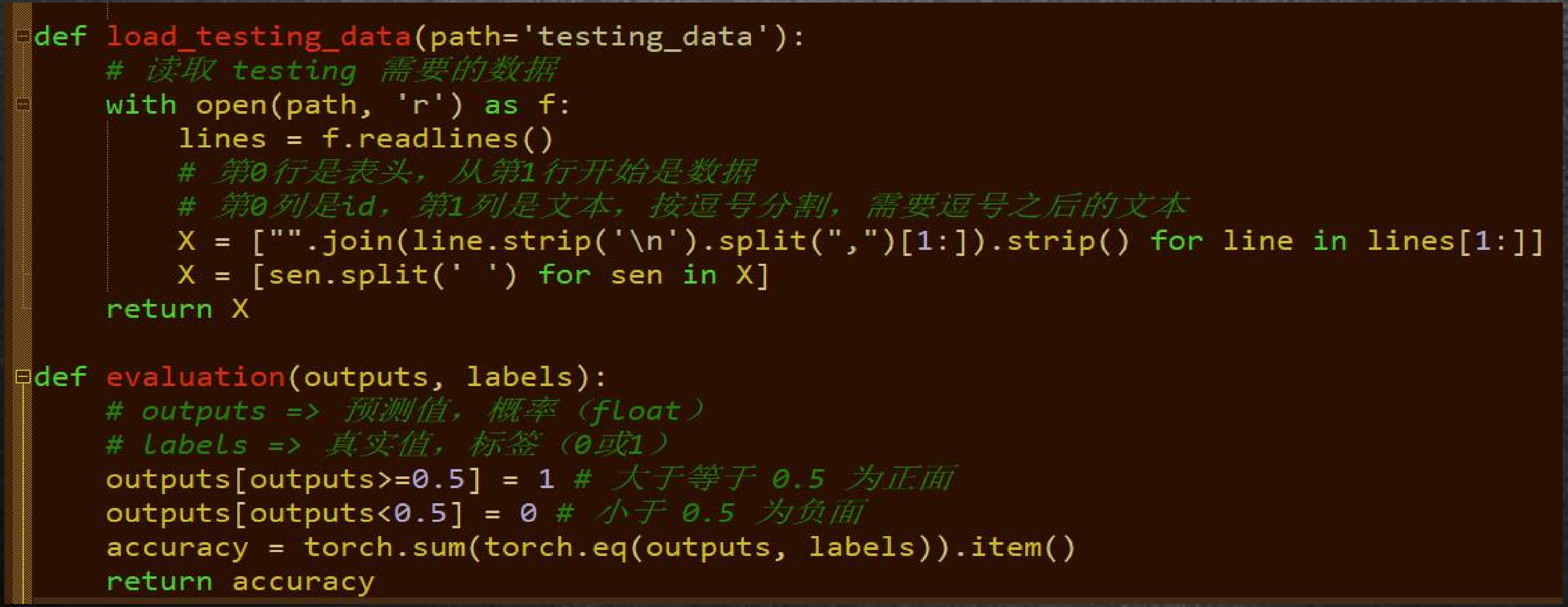
```
mkhang mlbo . dami niang followers ee . di q rin naman sia masisisi . desperate n kng desperate , pero dpt tlga replyn nia q = d
don ' t you hate it when you hang on to a seemingly interesting movie to see the ending only to find out that the ending sucks ?
ok so never went to the movies because friend wasn ' t feeling well but next weekend . back to work today , wasn ' t too bad .
can ' t wait to see diversity ' s performance !
i love britney spears haha joey this is what u do go party with eric or do things haha
wish i could call in but i can ' t do blogtalk from work
1 more day !
nursing celeste with a tummy ache .
hates being this burnt !! ouch
just couldn ' t sleep last night . working 7a 3p , than dinner with megan . happy bday jl !
i love slaves ! by david raccah , linkedin , rotfl
```



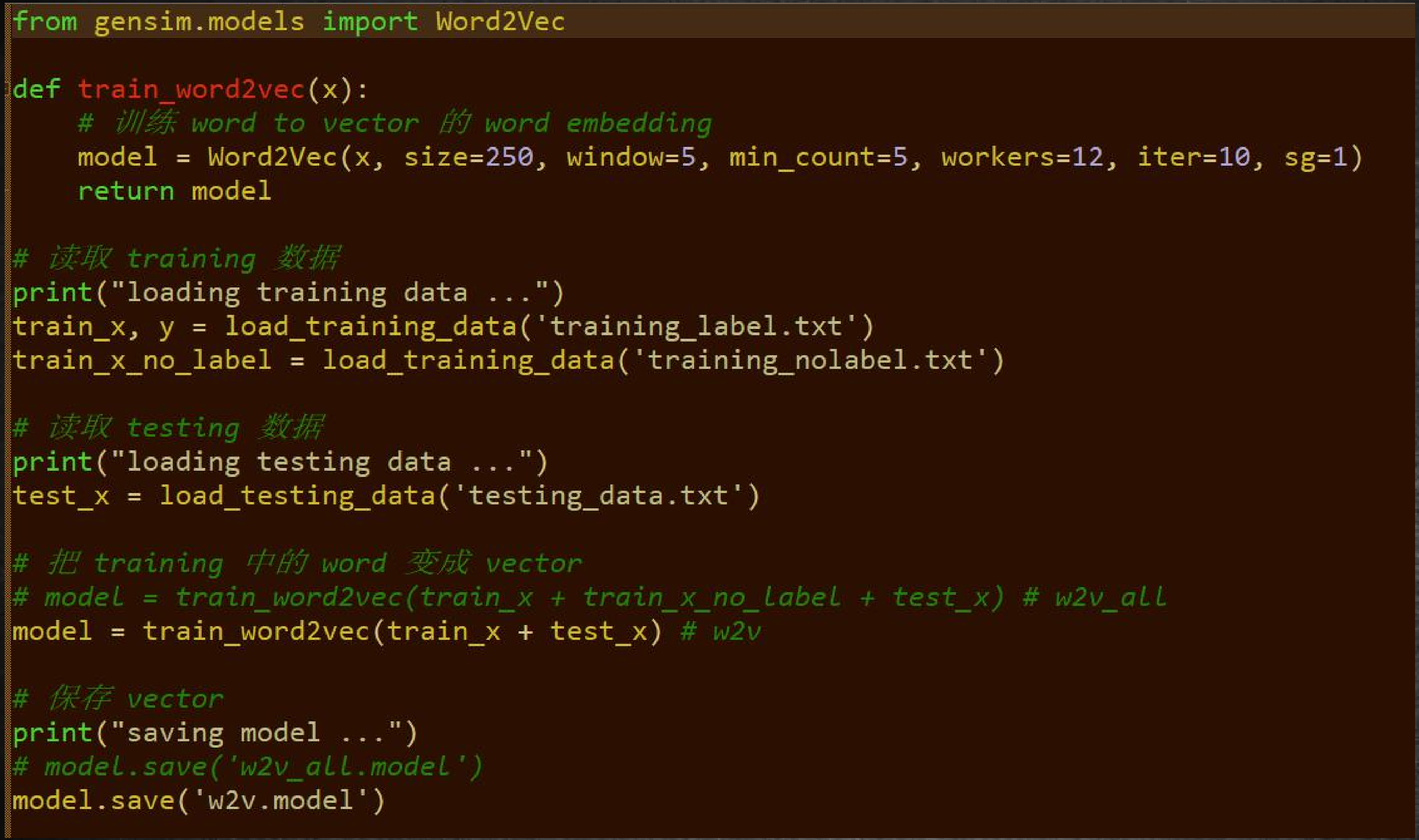
```
id,text
0,my dog ate our dinner . no , seriously ... he ate it .
1,omg last day sooon n of primary noooooo x im gona be swimming out of school wif the amount of tears am gona cry
2,stupid boys .. they ' re so .. stupid !
3,hi ! do u know if the nurburgring is open for tourists today ? we want to go , but there is an event today
4,having lunch in the office , and thinking of how to resolve this discount form issue
5,shopping was fun
6,wondering where all the nice weather has gone .
7,morning ! yeeesssssss new mimi in aug
8,umm ... maybe that ' s how the british spell it ?
9,yes it ' s 3 : 50 am . yes i ' m still awake . yes i can ' t sleep . yes i ' ll regret it tomorrow . haha i love you mr saturday
10,cute heart shaped portal cube . my baby is playing games , im reading fan fictions !
```



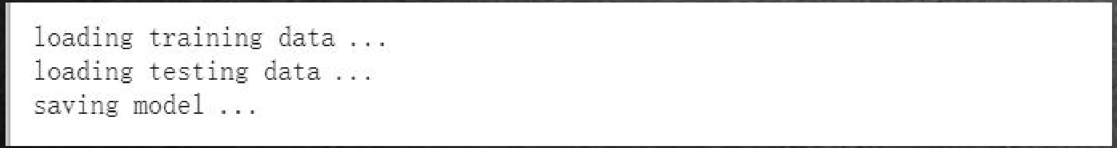
```python
def load_training_data(path='training_label.txt'):
    # 读取 training 需要的数据
    # 如果是 'training_label.txt'，需要读取 label，如果是 'training_nolabel.txt'，不需要读取 label
    if 'training_label' in path:
        with open(path, 'r') as f:
            lines = f.readlines()
            # lines是二维数组，第一维是行line(按回车分割)，第二维是每行的单词(按空格分割)
            lines = [line.strip('\n').split(' ') for line in lines]
        # 每行按空格分割后，第2个符号之后都是句子的单词
        x = [line[2:] for line in lines]
        # 每行按空格分割后，第0个符号是label
        y = [line[0] for line in lines]
        return x, y
    else:
        with open(path, 'r') as f:
            lines = f.readlines()
            # lines是二维数组，第一维是行line(按回车分割)，第二维是每行的单词(按空格分割)
            x = [line.strip('\n').split(' ') for line in lines]
        return x
```



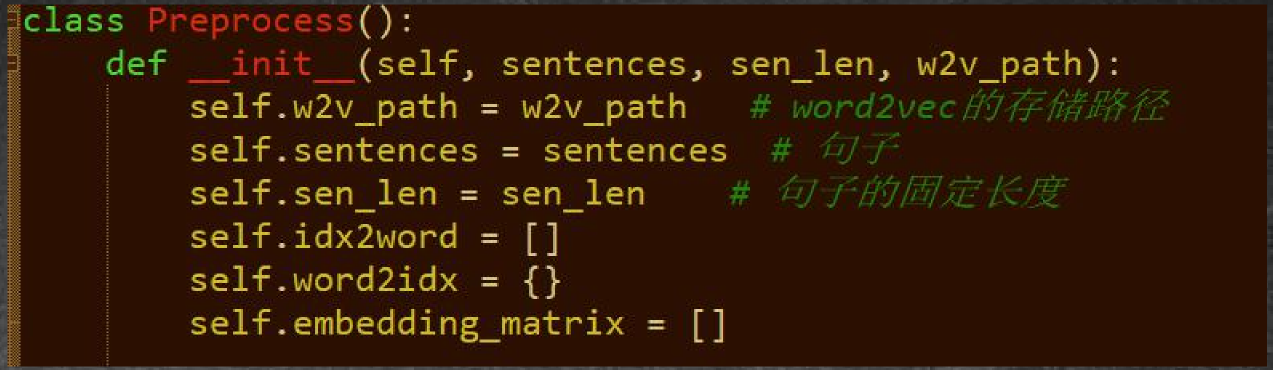
```python
def load_testing_data(path='testing_data'):
    # 读取 testing 需要的数据
    with open(path, 'r') as f:
        lines = f.readlines()
        # 第0行是表头，从第1行开始是数据
        # 第0列是id，第1列是文本，按逗号分割，需要逗号之后的文本
        X = ["".join(line.strip('\n').split(",")[1:]).strip() for line in lines[1:]]
        X = [sen.split(' ') for sen in X]
    return X

def evaluation(outputs, labels):
    # outputs => 预测值，概率（float）
    # labels => 真实值，标签（0或1）
    outputs[outputs>=0.5] = 1 # 大于等于 0.5 为正面
    outputs[outputs<0.5] = 0 # 小于 0.5 为负面
    accuracy = torch.sum(torch.eq(outputs, labels)).item()
    return accuracy
```



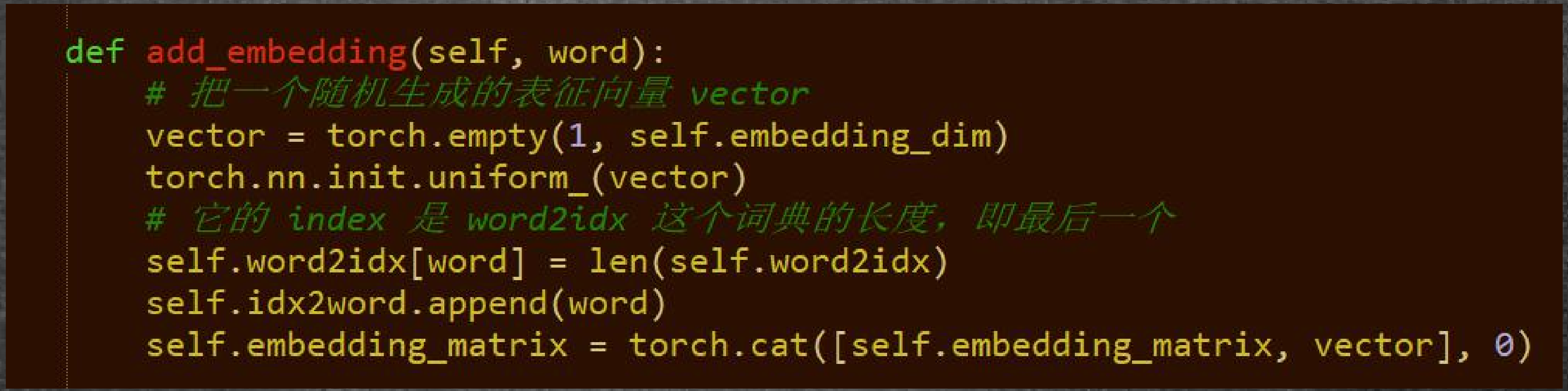
```python
from gensim.models import Word2Vec

def train_word2vec(x):
    # 训练 word to vector 的 word embedding
    model = Word2Vec(x, size=250, window=5, min_count=5, workers=12, iter=10, sg=1)
    return model

# 读取 training 数据
print("loading training data ...")
train_x, y = load_training_data('training_label.txt')
train_x_no_label = load_training_data('training_nolabel.txt')

# 读取 testing 数据
print("loading testing data ...")
test_x = load_testing_data('testing_data.txt')

# 把 training 中的 word 变成 vector
# model = train_word2vec(train_x + train_x_no_label + test_x) # w2v_all
model = train_word2vec(train_x + test_x) # w2v

# 保存 vector
print("saving model ...")
# model.save('w2v_all.model')
model.save('w2v.model')
```



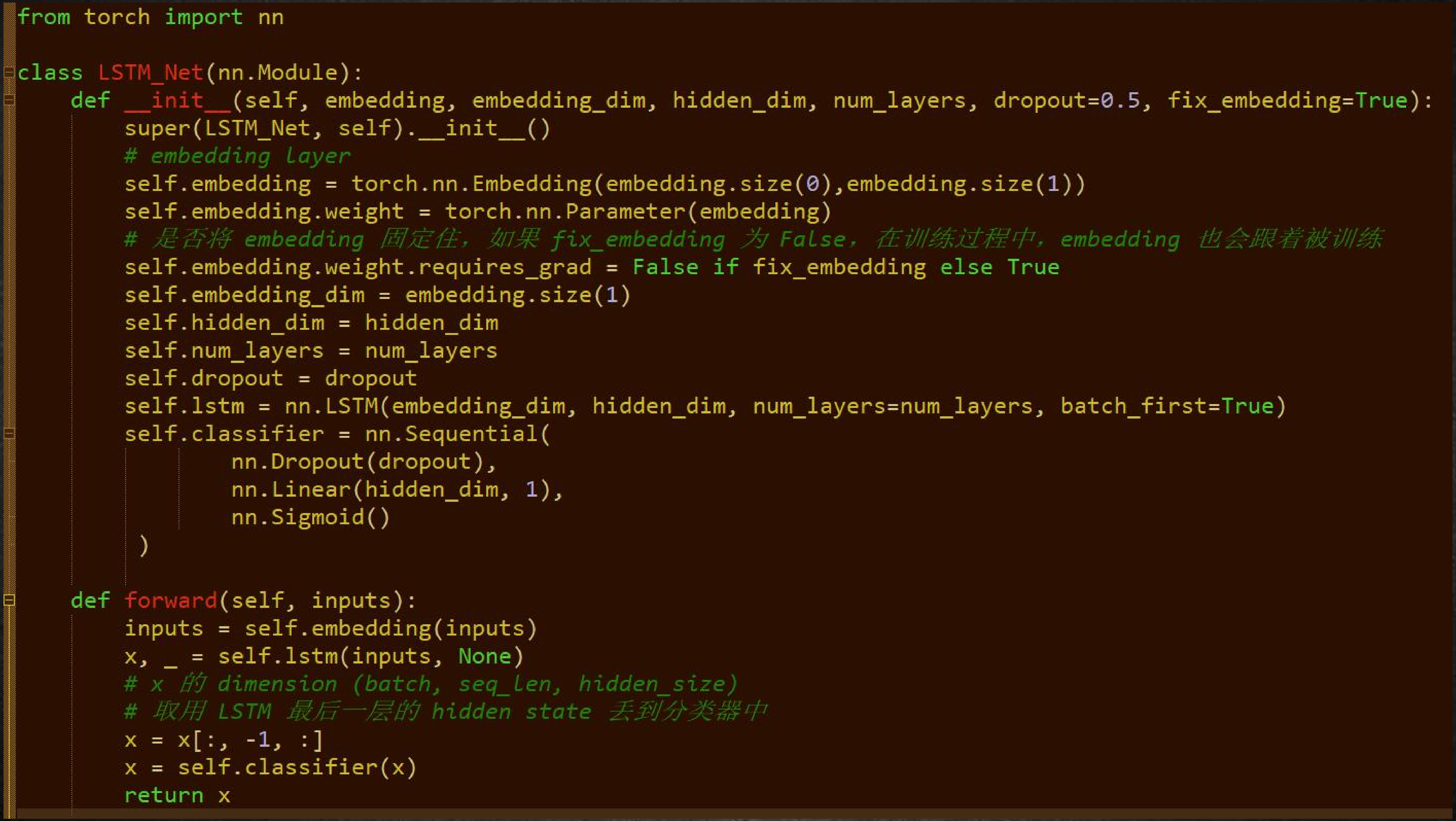
```
loading training data ...
loading testing data ...
saving model ...
```



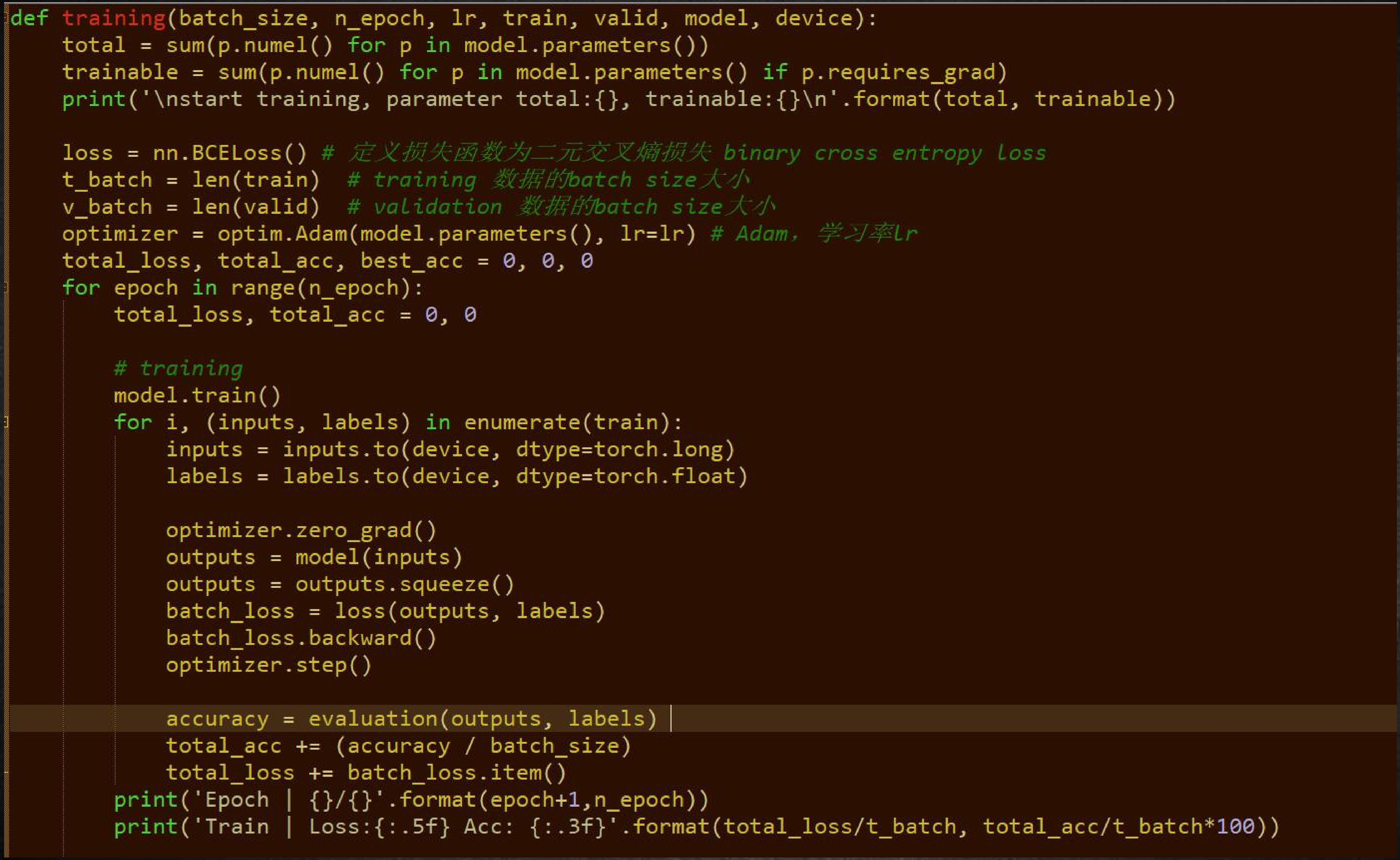
```python
class Preprocess():
    def __init__(self, sentences, sen_len, w2v_path):
        self.w2v_path = w2v_path   # word2vec的存储路径
        self.sentences = sentences  # 句子
        self.sen_len = sen_len    # 句子的固定长度
        self.idx2word = []
        self.word2idx = {}
        self.embedding_matrix = []
```



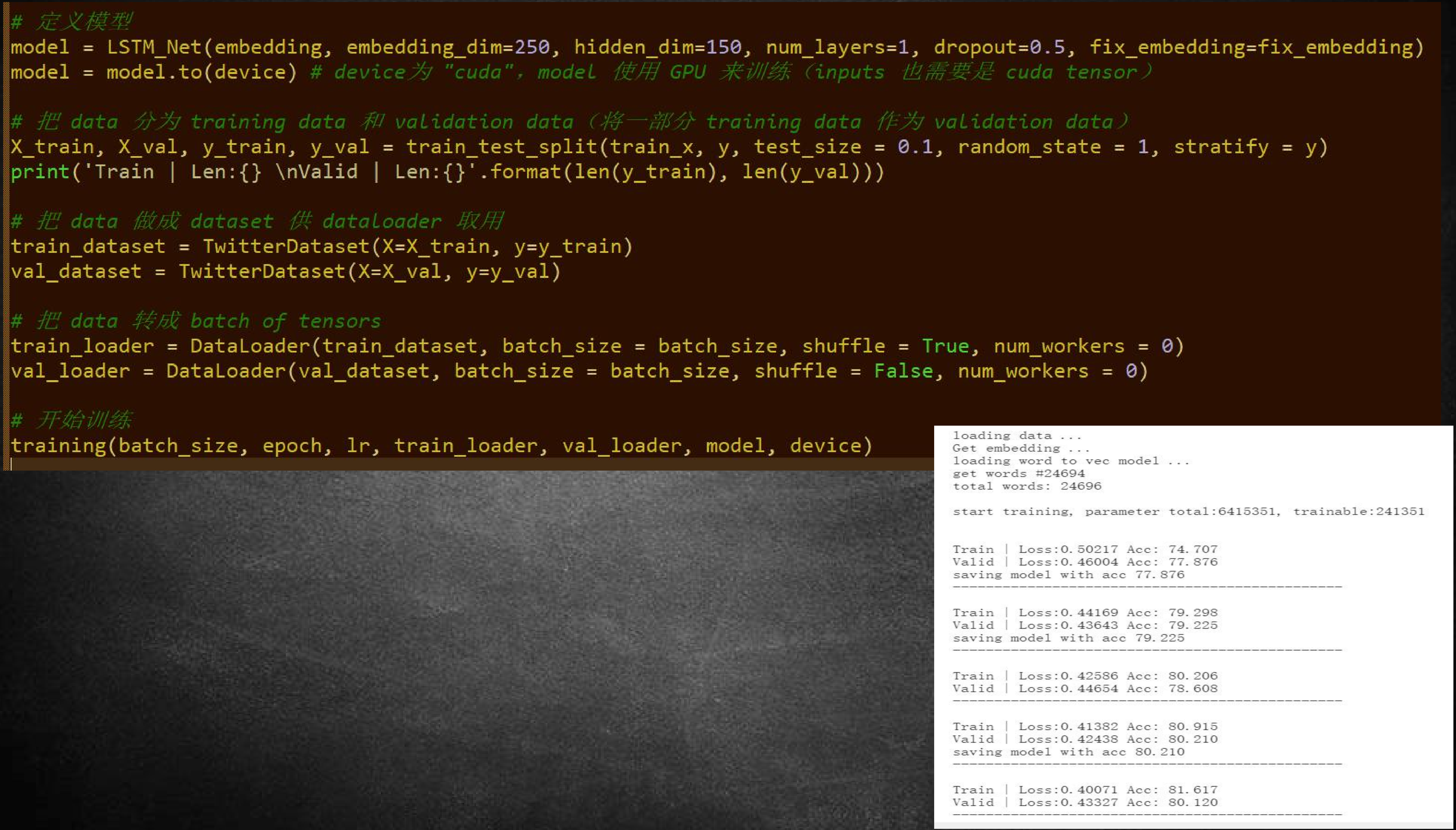
```python
def add_embedding(self, word):
    # 把一个随机生成的表征向量 vector
    vector = torch.empty(1, self.embedding_dim)
    torch.nn.init.uniform_(vector)
    # 它的 index 是 word2idx 这个词典的长度，即最后一个
    self.word2idx[word] = len(self.word2idx)
    self.idx2word.append(word)
    self.embedding_matrix = torch.cat([self.embedding_matrix, vector], 0)
```



```python
from torch import nn

class LSTM_Net(nn.Module):
    def __init__(self, embedding, embedding_dim, hidden_dim, num_layers, dropout=0.5, fix_embedding=True):
        super(LSTM_Net, self).__init__()
        # embedding layer
        self.embedding = torch.nn.Embedding(embedding.size(0),embedding.size(1))
        self.embedding.weight = torch.nn.Parameter(embedding)
        # 是否将 embedding 固定住，如果 fix_embedding 为 False，在训练过程中，embedding 也会跟着被训练
        self.embedding.weight.requires_grad = False if fix_embedding else True
        self.embedding_dim = embedding.size(1)
        self.hidden_dim = hidden_dim
        self.num_layers = num_layers
        self.dropout = dropout
        self.lstm = nn.LSTM(embedding_dim, hidden_dim, num_layers=num_layers, batch_first=True)
        self.classifier = nn.Sequential(
                nn.Dropout(dropout),
                nn.Linear(hidden_dim, 1),
                nn.Sigmoid()
         )

    def forward(self, inputs):
        inputs = self.embedding(inputs)
        x, _ = self.lstm(inputs, None)
        # x 的 dimension (batch, seq_len, hidden_size)
        # 取用 LSTM 最后一层的 hidden state 丢到分类器中
        x = x[:, -1, :]
        x = self.classifier(x)
        return x
```

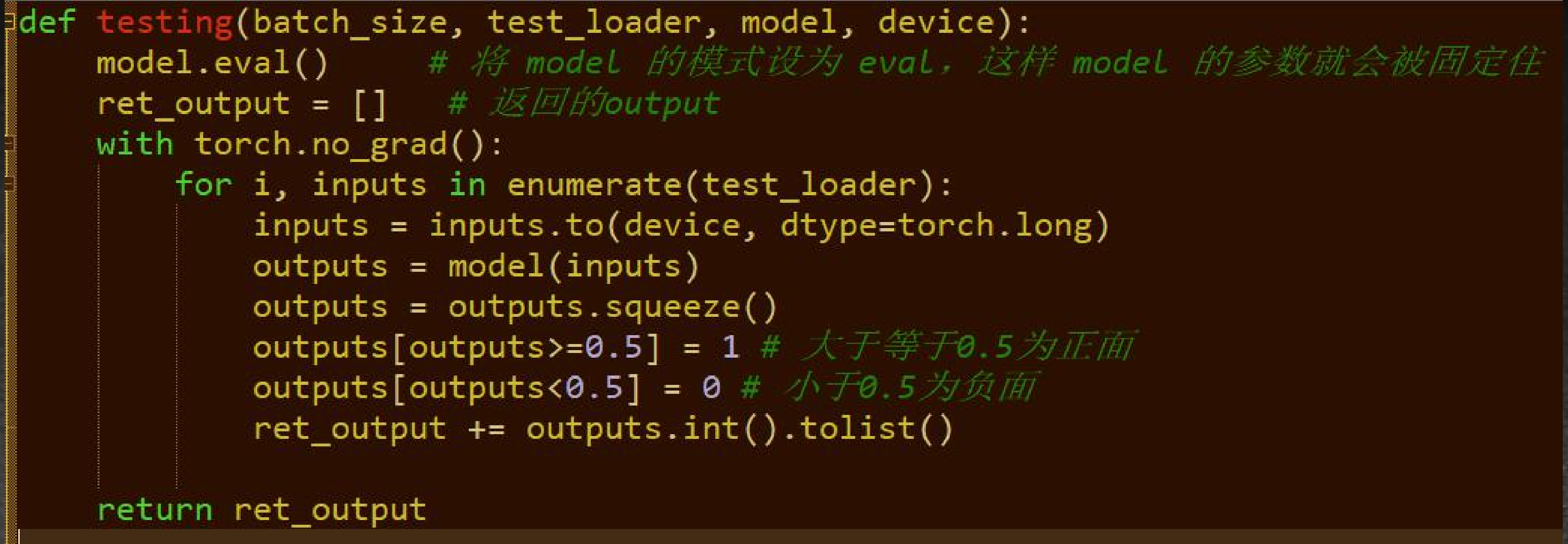


```
def training(batch_size, n_epoch, lr, train, valid, model, device):
    total = sum(p.numel() for p in model.parameters())
    trainable = sum(p.numel() for p in model.parameters() if p.requires_grad)
    print('\nstart training, parameter total:{}, trainable:{}\n'.format(total, trainable))

    loss = nn.BCELoss() # 定义损失函数为二元交叉熵损失 binary cross entropy loss
    t_batch = len(train)  # training 数据的batch size大小
    v_batch = len(valid)  # validation 数据的batch size大小
    optimizer = optim.Adam(model.parameters(), lr=lr) # Adam，学习率lr
    total_loss, total_acc, best_acc = 0, 0, 0
    for epoch in range(n_epoch):
        total_loss, total_acc = 0, 0

        # training
        model.train()
        for i, (inputs, labels) in enumerate(train):
            inputs = inputs.to(device, dtype=torch.long)
            labels = labels.to(device, dtype=torch.float)

            optimizer.zero_grad()
            outputs = model(inputs)
            outputs = outputs.squeeze()
            batch_loss = loss(outputs, labels)
            batch_loss.backward()
            optimizer.step()

            accuracy = evaluation(outputs, labels)
            total_acc += (accuracy / batch_size)
            total_loss += batch_loss.item()
        print('Epoch | {}/{}'.format(epoch+1,n_epoch))
        print('Train | Loss:{:.5f} Acc: {:.3f}'.format(total_loss/t_batch, total_acc/t_batch*100))
```

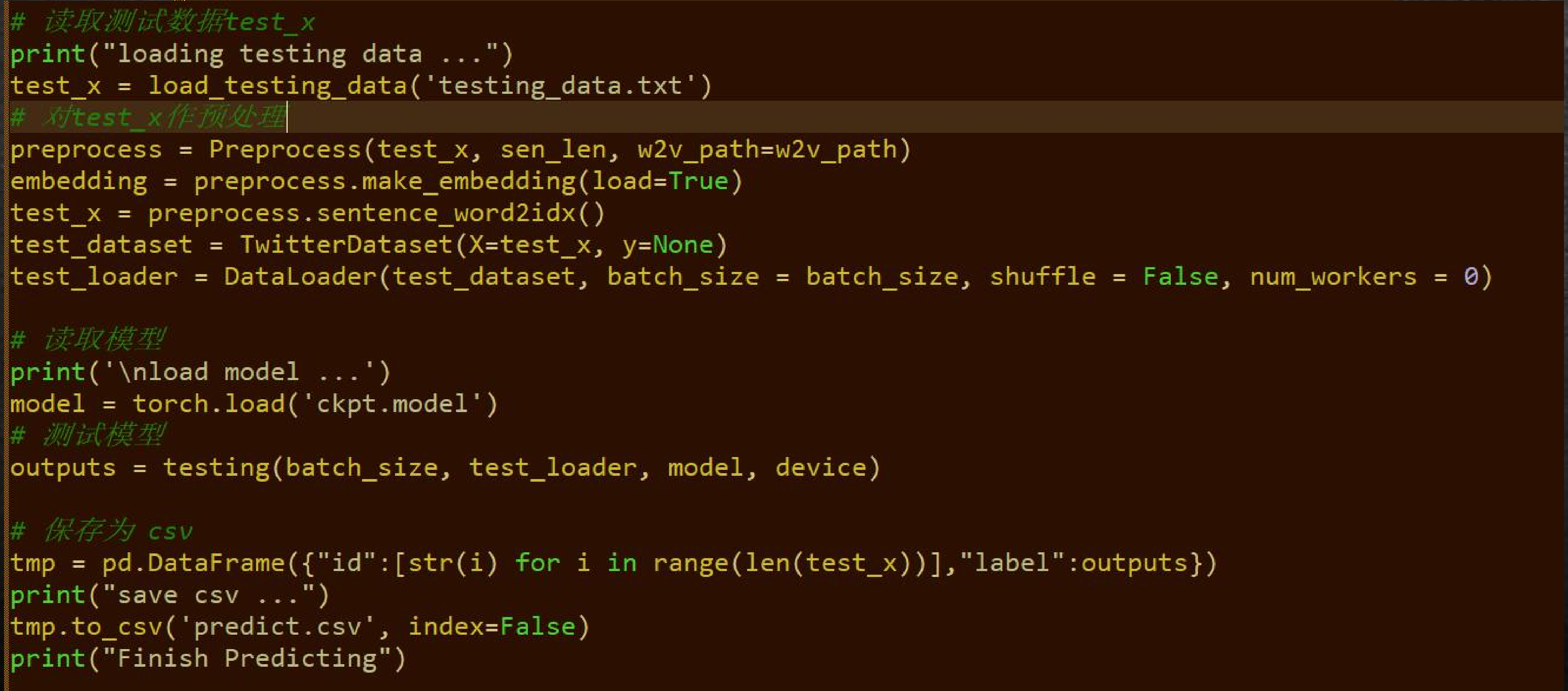


```python
# 定义模型
model = LSTM_Net(embedding, embedding_dim=250, hidden_dim=150, num_layers=1, dropout=0.5, fix_embedding=fix_embedding)
model = model.to(device) # device为 "cuda"，model 使用 GPU 来训练（inputs 也需要是 cuda tensor）

# 把 data 分为 training data 和 validation data（将一部分 training data 作为 validation data）
X_train, X_val, y_train, y_val = train_test_split(train_x, y, test_size = 0.1, random_state = 1, stratify = y)
print('Train | Len:{} \nValid | Len:{}'.format(len(y_train), len(y_val)))

# 把 data 做成 dataset 供 dataloader 取用
train_dataset = TwitterDataset(X=X_train, y=y_train)
val_dataset = TwitterDataset(X=X_val, y=y_val)

# 把 data 转成 batch of tensors
train_loader = DataLoader(train_dataset, batch_size = batch_size, shuffle = True, num_workers = 0)
val_loader = DataLoader(val_dataset, batch_size = batch_size, shuffle = False, num_workers = 0)

# 开始训练
training(batch_size, epoch, lr, train_loader, val_loader, model, device)
```

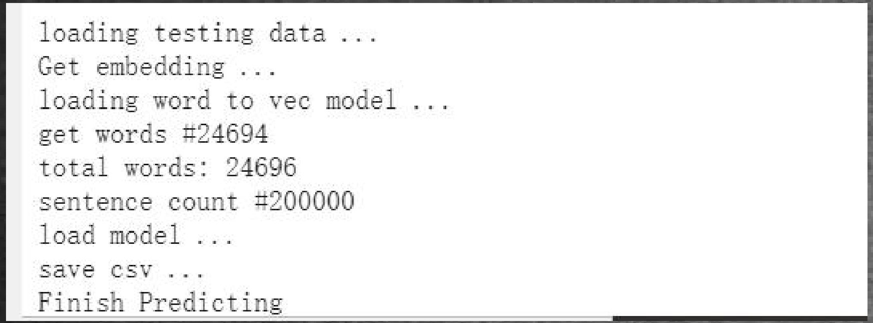
```
loading data ...
Get embedding ...
loading word to vec model ...
get words #24694
total words: 24696

start training, parameter total:6415351, trainable:241351


Train | Loss:0.50217 Acc: 74.707
Valid | Loss:0.46004 Acc: 77.876
saving model with acc 77.876
-----------------------------------------------

Train | Loss:0.44169 Acc: 79.298
Valid | Loss:0.43643 Acc: 79.225
saving model with acc 79.225
-----------------------------------------------

Train | Loss:0.42586 Acc: 80.206
Valid | Loss:0.44654 Acc: 78.608
-----------------------------------------------

Train | Loss:0.41382 Acc: 80.915
Valid | Loss:0.42438 Acc: 80.210
saving model with acc 80.210
-----------------------------------------------

Train | Loss:0.40071 Acc: 81.617
Valid | Loss:0.43327 Acc: 80.120
-----------------------------------------------
```



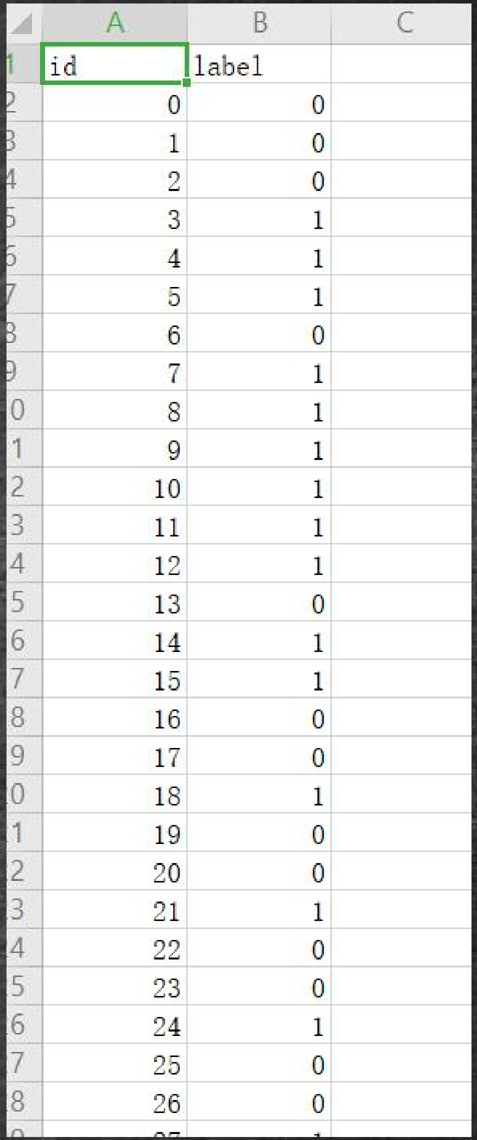
```python
def testing(batch_size, test_loader, model, device):
    model.eval()     # 将 model 的模式设为 eval，这样 model 的参数就会被固定住
    ret_output = []   # 返回的output
    with torch.no_grad():
        for i, inputs in enumerate(test_loader):
            inputs = inputs.to(device, dtype=torch.long)
            outputs = model(inputs)
            outputs = outputs.squeeze()
            outputs[outputs>=0.5] = 1 # 大于等于0.5为正面
            outputs[outputs<0.5] = 0 # 小于0.5为负面
            ret_output += outputs.int().tolist()

    return ret_output
```



```python
# 读取测试数据test_x
print("loading testing data ...")
test_x = load_testing_data('testing_data.txt')
# 对test_x作预处理
preprocess = Preprocess(test_x, sen_len, w2v_path=w2v_path)
embedding = preprocess.make_embedding(load=True)
test_x = preprocess.sentence_word2idx()
test_dataset = TwitterDataset(X=test_x, y=None)
test_loader = DataLoader(test_dataset, batch_size = batch_size, shuffle = False, num_workers = 0)

# 读取模型
print('\nload model ...')
model = torch.load('ckpt.model')
# 测试模型
outputs = testing(batch_size, test_loader, model, device)

# 保存为 csv
tmp = pd.DataFrame({"id":[str(i) for i in range(len(test_x))],"label":outputs})
print("save csv ...")
tmp.to_csv('predict.csv', index=False)
print("Finish Predicting")
```



```
loading testing data ...
Get embedding ...
loading word to vec model ...
get words #24694
total words: 24696
sentence count #200000
load model ...
save csv ...
Finish Predicting
```



id	label
0	0
1	0
2	0
3	1
4	1
5	1
6	0
7	1
8	1
9	1
10	1
11	1
12	1
13	0
14	1
15	1
16	0
17	0
18	1
19	0
20	0
21	1
22	0
23	0
24	1
25	0
26	0