Tesseract v5.0 +Mnist LSTM训练

博客链接：<https://blog.csdn.net/weixin_42914339/article/details/108171299>

项目Github链接：

项目码云链接：<https://gitee.com/zkyyun/digit_mnist_ocr>

平台及版本：Win10+Tesseract v5.0

注：官方说法是4.0开始支持lstm，但是安装4.0后在config文件夹下并没有lstmbox。makebox（v3.0训练时生成box的配置文件）生成的box文件格式又与lstmbox不同，因此选择装5.0。

Tesseract训练数据本应是白底黑字，小编开始时没考虑训练了黑底白字的，大多数图片也是黑底的，懒得再截一次。测试部分也对比了黑底、白底训练效果的差异。

目录

[一、Tesseract安装及jTessBoxEditor下载 2](#_Toc28612)

[二、开始项目 2](#_Toc32226)

[三、主文件夹说明 2](#_Toc16683)

[四、项目总操作步骤 3](#_Toc19287)

[1. creat\_data文件夹下操作（获取数据） 3](#_Toc6633)

[2. data\_merge文件夹下操作（合并数据） 3](#_Toc27489)

[（1）获取合并数据及合并数据数字序列字典 3](#_Toc29218)

[（2）合成tif 3](#_Toc5202)

[（3）命令行生成box（ mergeTif/目录下运行） 4](#_Toc21142)

[（4）对box文件矫正（三种方法） 4](#_Toc4287)

[（5）命令行生成lstmf文件用于训练/测试（ mergeTif/目录下运行） 5](#_Toc29969)

[3. train文件夹下操作（训练） 5](#_Toc3753)

[（1）从已有.traineddata文件中提取.lstm文件用于当训练起点 5](#_Toc17772)

[（2）创建写有 .lstmf文件路径的txt文件（train/目录下执行） 5](#_Toc20720)

[（3）训练（train/目录下执行） 6](#_Toc12207)

[（4）合并模型（train/目录下执行） 6](#_Toc7554)

[（5）完成，开始测试 6](#_Toc19464)

[五.总结 8](#_Toc21988)

[1.随机序列问题 8](#_Toc16311)

[2.命令行创建txt文本问题 9](#_Toc21756)

[参考链接 9](#_Toc8736)

**一、Tesseract安装及jTessBoxEditor下载**

参考：<https://blog.csdn.net/qq_31112205/article/details/100159963>

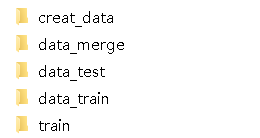
本项目链接中也有对应安装包。

Python使用还需要安装pytesseract模块：pip install pytesseract

**二、开始项目**

注：本教程同时处理训练样本和测试样本

首先打开主文件夹目录



-----------------------------------------------------------------------------------

**三、主文件夹说明**

creat\_data 创建数据文件夹，从mnist数据集获取数据到data\_test 、data\_train

data\_test 测试集数据文件夹，数据由creat\_data获取

data\_train 训练集数据文件夹，数据由creat\_data获取

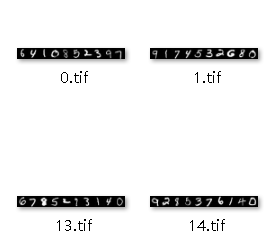
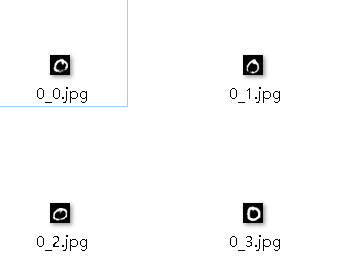
data\_merge 合并数据文件夹，负责将图像合成多数字图像，并最终合成 tif

train 训练文件夹，进行训练相关操作

-----------------------------------------------------------------------------------

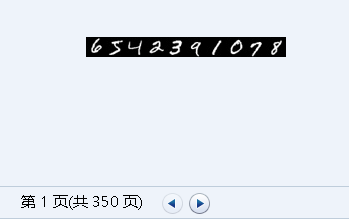
注：合并数据、最终合成tif说明

creat\_data生成的data\_test/文件夹数据如下左图，每张图片只有一个数字



data\_merge中merge\_img.py将每10张图片以随机序列合成一张（10个数字），如下图。随机序列是为了使数据多元化。

最终合成tif由jTessBoxEditor将以上合成图合为一张，一张tif里包含多张图



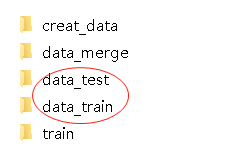
**四、项目总操作步骤**

1. **creat\_data文件夹下操作**（获取数据）

获取图像数据，分为训练数据和测试数据，这里前350训练，后150测试：

使用creat\_data/creat\_data.py

将在主文件夹下创建数据文件夹（data\_test、data\_train）

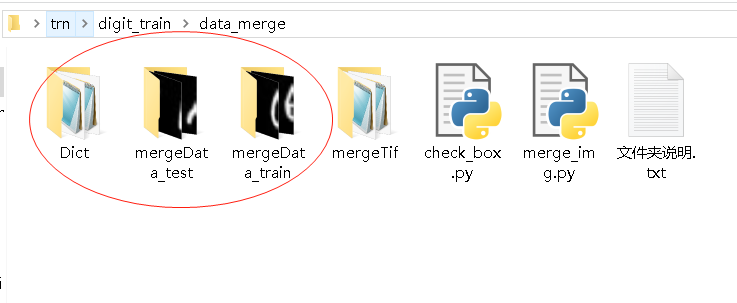


1. **data\_merge文件夹下操作**（合并数据）

（1）获取合并数据及合并数据数字序列字典

使用data\_merge/merge\_img.py （合成的数字序列是随机的，防止训练数据单一）

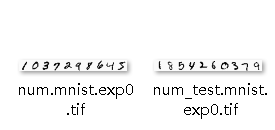
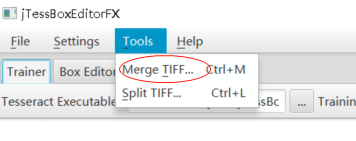
将创建 合并数据到目录（mergeData\_train/、mergeData\_test/），字典在目录（Dict/）



注：字典用于判断正确率及纠正box文件

（2）合成tif

使用jTessBoxEditor将①获得的tif合成，命名num[\_test].mnist.exp0.tif，放文件夹 mergeTif下

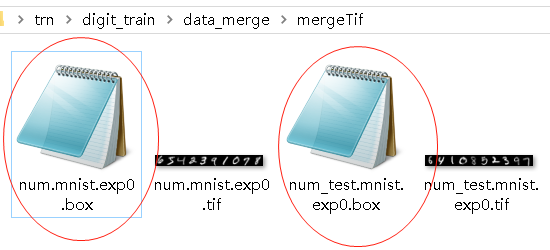


注：合成tif命名规则：<语言名.字体名.版本号>。

（3）命令行生成box（ mergeTif/目录下运行）

tesseract num.mnist.exp0.tif num.mnist.exp0 -l eng --psm 6 lstmbox

tesseract num\_test.mnist.exp0.tif num\_test.mnist.exp0 -l eng --psm 6 lstmbox



（4）对box文件矫正（三种方法）

①jTessBoxEditor矫正

网上教程看到的好像都是这个，老实一个一个手改？手写数字基本没有识别出来的，是不是有10万个训练数据就得改10万个...小编最后要做的是手写汉字识别，那岂不是...

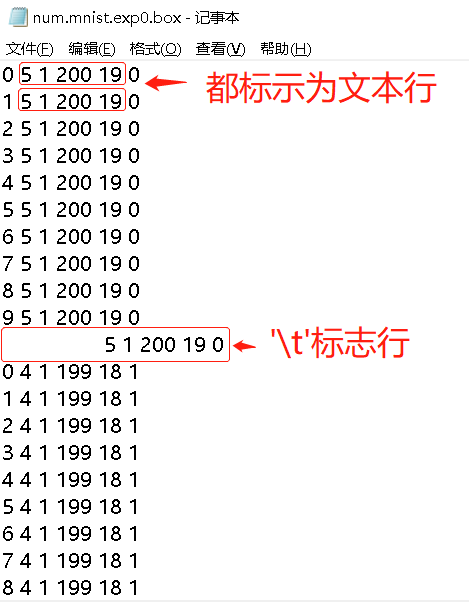
②直接打开box文件矫正

基于（1）的情况，小编尝试打开box文件，发现记事本也可以正常读取。直接修改比 jTessBoxEditor修改方便100倍啊

如图，box文件中所有文字的文本框都为整行，后面跟一个 \t标志行。每行的首个字符即识别出的字符，接下来4个参数为文本框的左上角坐标（x,y）及宽高（w,h）。如第一行识别出字符为0，框（5，1，200，19）。

我们只需要修改第一个字符为正确字符，例如第一张图像序列为0123456789，后接一 ’\t’标志行，有多余行则删，少行则添。

这种方法在小编第一次尝试训练时候使用，由于第一次尝试，所有图片的数字序列都定为0123456789，修改相对容易。虽然只有50张图，但是box文件修改完也有50\*11=550 行，累



③程序矫正 check\_box.py

基于（2）中发现box文件的格式，以及2①中可以获得图像的数字序列字典，小编将其结合，让程序完成自动校对。大概...方便100000000000000000倍吧

check\_box.py 中将原box文件命名为old\_[boxName].box，并新建一个原名的文件用于保存正确box，方便后边操作。

（5）命令行生成lstmf文件用于训练/测试（ mergeTif/目录下运行）

cd .. & md lstmf & cd mergeTif

tesseract num.mnist.exp0.tif ../lstmf/num.mnist.exp0 -l eng --psm 6 lstm.train

tesseract num\_test.mnist.exp0.tif ../lstmf/num\_test.mnist.exp0 -l eng --psm 6 lstm.train

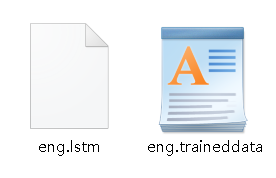
注：文件夹说明见=> data\_merge/文件夹说明.txt

1. **train文件夹下操作**（训练）

（1）从已有.traineddata文件中提取.lstm文件用于当训练起点

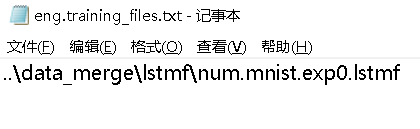
从https://github.com/tesseract-ocr/tessdata\_best下载对应语言，例如这里是eng则下载 eng.traineddata到train/文件夹下，并在该文件夹下执行命令行进行提取：

combine\_tessdata -e eng.traineddata eng.lstm



（2）创建写有 .lstmf文件路径的txt文件（train/目录下执行）

命令行：set /p =..\data\_merge\lstmf\num.mnist.exp0.lstmf<nul>eng.training\_files.txt



（3）训练（train/目录下执行）

训练命令行：

md output

lstmtraining --debug\_interval -1 --max\_iterations 10000 --target\_error\_rate 0.01 --continue\_from=".\eng.lstm" --model\_output=".\output\output" --train\_listfile=".\eng.training\_files.txt" --traineddata=".\eng.traineddata"

参数意义：

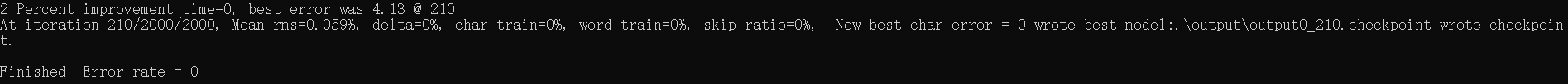
--debug\_interval 当值为-1时，训练结束，会显示训练的一些结果参数，此参数可略去。

--max\_iterations 指明训练遍历次数。还可以指定参数：--target\_error\_rate 0.01 训练至错误率低于0.01%终止。

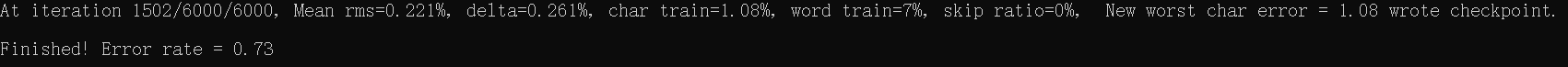
--continue\_from 训练从哪里开始，这里指定从eng.lstm文件。也可从训练生成的阶段文件output\_checkpoint开始。

--model\_output 模型训练输出的路径  
 --train\_listfile 指定上一步创建的txt文件路径。  
 --traineddata 第1步中下载的.traineddata文件的路径。

随着训练均方根、Δ、train（错误率）都会下降，结束会显示best model，finish



若迭代次数太少，则结果达不到要求的错误率，显示如下:



出现以上情况可再从最新检查点开始训练：把上面命令行中--continue\_from=".\eng.lstm"改为--continue\_from="output\_checkpoint"，同时增大--max\_iterations，例如本项目中需要10000

（4）合并模型（train/目录下执行）

将checkpoint文件和.traineddata文件合并成新的.traineddata文件（若最佳模型不是output\_checkpoint，命令行中改为最佳模型文件名）命令行：

lstmtraining --stop\_training --traineddata=".\eng.traineddata" --continue\_from=".\output\output\_checkpoint" --model\_output=".\output\mnist.traineddata"

参数意义：

–stop\_training 默认要有

–traineddata 第1步中下载的已有.traineddata文件的路径

–continue\_from 上一步生成的 最佳模型文件路径

–model\_output 输出的路径和文件名

（5）完成，开始测试

两种测试方式，1.利用测试集进行测试，2.将新生成的语言包导入安装路径进行实际的文字识别。

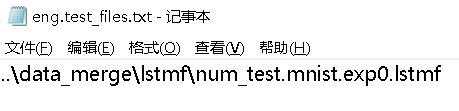
------------------------------

\*测试方式1：测试集测试

①生成测试集.lstmf文件，前边已生成到 data\_merge\lstmf\num\_test.mnist.exp0.lstmf

②创建写有 .lstmf文件路径的txt文件（train/目录下执行）：

命令行：set /p =..\data\_merge\lstmf\num\_test.mnist.exp0.lstmf<nul>eng.test\_files.txt



③测试（train/目录下执行）

lstmeval --model=".\output\mnist.traineddata" --eval\_listfile=".\eng.test\_files.txt"

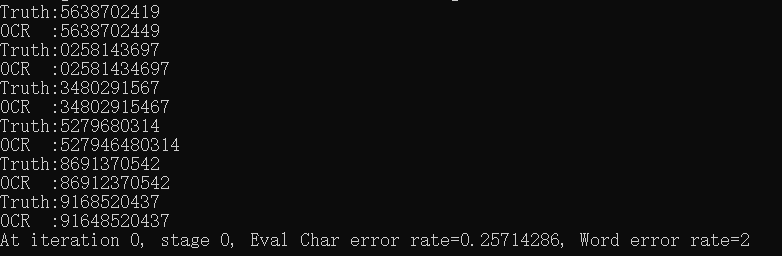
参数意义：

--model ： 生成的语言包路径  
--eval\_listfile： 刚刚创建的txt文件

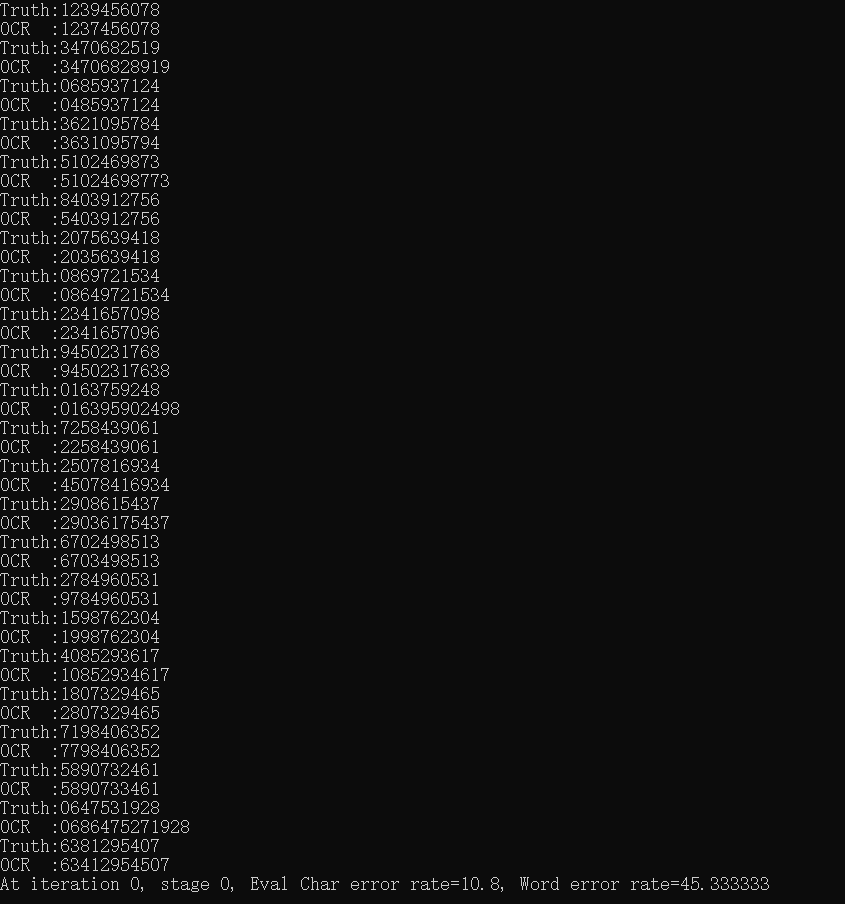
④测试结果

[黑底白字测试结果

训练集测试结果：



测试集测试结果：

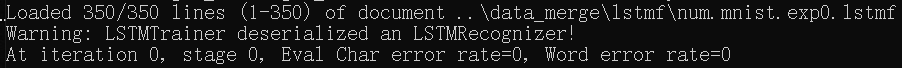


可见训练集测试效果比较好，350张图只有6张图出现错误；测试集150张图有67张图有错误。另上述测试中tesseract遇到识别结果有多个可能的会都打印输出，如下图中1的识别结果有1、7两种可能。

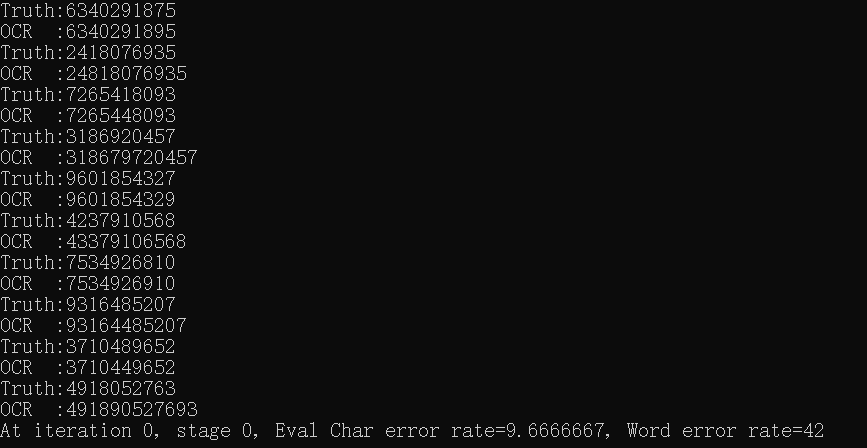


[白底黑字测试结果

训练集测试结果：



测试集测试结果：



效果相比黑底白字好。训练集测试达到100%正确率；测试集150张图有62张图有错误。

注：mnist有一些数字真挺难辨别，不知训练中是否可以调整哪些参数来改进训练效 果，欢迎大佬们提供建议

\*测试方法2，pytesseract测试

①将生成的语言包 .traineddata复制到Tesseract安装目录的tessdata下

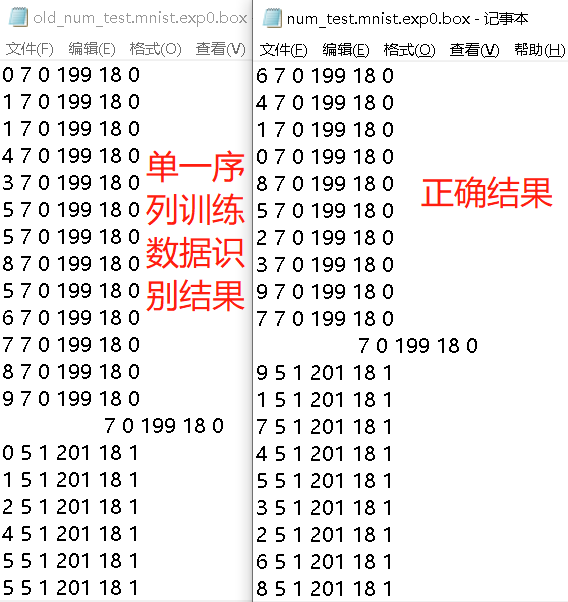
②使用pytesseract测试

**五.总结**

1.随机序列问题

在处理数字序列时候随机组合，可以增加数据多样性。若所有序列都是0123456789，则错划分可能性大，若为随机序列，1的旁边可能是3可能是4，错划分可能性更低。小编最终做的是汉字识别，同一序列划分出错应该会更常见。

如下为第一次测试数据，训练数据为50组，序列都是0123456789，识别效果非常差。

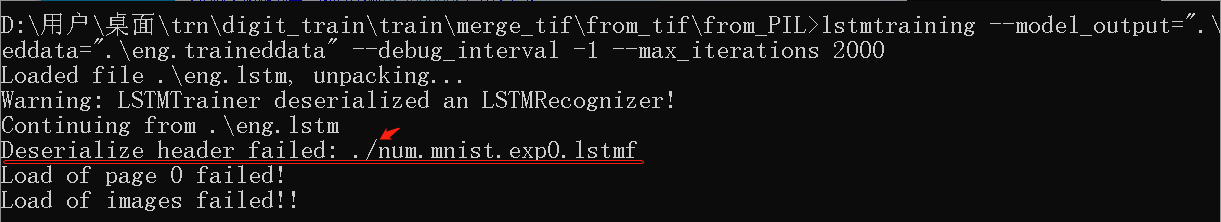


2.命令行创建txt文本问题

在步骤3（2）中创建txt文件原采用下面命令行：

echo ..\data\_merge\lstmf\num.mnist.exp0.lstmf>eng.training\_files.txt

训练时出现如下问题（这是第一次训练时候的截图，路径和本文不同），即找不到 num.mnist.exp0.lstmf。



训练命令行（见下文）中是从eng.training\_files.txt文件中读取该文件路径的，所以问题就在txt。打开发现多出来一空行，删除后训练成功运行。

多出来空行是因为echo 在输出后会自动换行，类似python中print()，如果非要使用，那么执行命令行后要手动删掉空行。新命令行 set /p 原是bat中用来创建外部输入变量的，类似python中input()，等号后边是对用户输入的提示内容，提示内容输出时不换行；<nul表示从空设备输入，set /p =...<nul 表示不需接受外部输入，因此set /p =...<nul 可代替echo做不换行输出使用。Bat教程可参考：<https://www.w3cschool.cn/dosmlxxsc1/wpyirh.html>

参考链接

Tesseract4.0+Mnist：<https://blog.csdn.net/ayayayayo/article/details/107811871>

官方lstm训练：<https://tesseract-ocr.github.io/tessdoc/TrainingTesseract-4.00>

官方提高识别品质：<https://tesseract-ocr.github.io/tessdoc/ImproveQuality>