1. 生成单字符png文件，并划分到各自文件夹，文件夹名即为字符名
2. 运行crop命令，对单字符png进行去除白边，并检查成果
3. 执行box\_gen命令，在同级别目录下生成同名lmstf文件
4. 通过已有traineddata文件分离得到lstm文件，需采用best模型，而非fast模型：combine\_tessdata -e eng.traineddata eng.lstm
5. 基于既有的lstm模型，通过continue\_from执行训练：lstmtraining --model\_output my\_model --continue\_from eng.lstm --traineddata eng.traineddata --train\_listfile listfile.txt --max\_iterations 4000
6. lstmtraining --stop\_training --continue\_from my\_model\_checkpoint --traineddata eng.traineddata --model\_output final.traineddata

in tesseract training, when I try to combine\_tessdata. It reports as failed to create a temporary file

ChatGPT

The "failed to create a temporary file" error usually points to a permission issue, space availability issue, or that the path specified doesn't exist. Here are some steps you can take to troubleshoot the issue:

Check directory and file permissions: Make sure that the user running the combine\_tessdata command has sufficient permission to write to the directory where the temporary file is supposed to be created. If it doesn't, you might need to change the permissions using the chmod command, or run the command as a user who has the necessary permissions (like root).

Check for sufficient disk space: If your disk is full, then creating a new file will fail. You can check the amount of free disk space with the df command on Unix-based systems, or using the properties option on the drive in Windows. If the disk is full, you'll need to delete some files to make space.

Verify the path: The path you specified might be incorrect or not exist. Make sure that the directory where you want to create the file exists and is correctly spelled.

Check Environment Variables: Sometimes, temp file creation is determined by certain environment variables (like TMPDIR, TEMP, or TMP on Unix-based systems). Check these variables and make sure they're set to directories that exist and can be written to.

If the issue still persists after checking all these, consider sharing the exact command you're using and any additional output. It could also be helpful to know what operating system you're using, as some solutions might be platform-specific.

User

command: combine\_tessdata -o eng.traineddata listfile.txt

ChatGPT

The combine\_tessdata command you're using seems correct, so the issue is likely due to other reasons I mentioned earlier. I'll provide a detailed step-by-step guide to debug this issue:

Check for write permissions in the current directory: Run ls -l in the terminal to check the permissions of your current directory. If you see that your user does not have write permissions, grant them by using the chmod command (you might need to use sudo if you're not the owner).

Check if there is enough disk space: Run df -h . to see if there is enough disk space in the current partition. If the Use% is 100% or near, you need to clean up some space.

Check the content of your listfile.txt: The file listfile.txt should contain the names of the .traineddata files (without the extension) you want to combine. Make sure these files exist in the current directory and your user has read permissions for these files.

Check your TEMP/TMPDIR environment variable: The TEMP or TMPDIR environment variable is used to determine the directory where temporary files are created. Ensure this is set to a directory that exists and where your user has write permissions. You can check this by running echo $TMPDIR or echo $TEMP.

Update Tesseract: If none of the above work, you might want to try updating your Tesseract software. It's possible that there's a bug in the version you're using that's causing this issue. You can do this by cloning the latest version of the Tesseract from the GitHub repo and then building it from source.

If you still can't resolve the issue, please provide any error messages you're getting or any other information about the steps you've taken. It would be helpful to see any additional output from the combine\_tessdata command, for example.

User

the windows 10 is used

ChatGPT

If you are using Windows, the permission issues are usually less common but they can still occur. Here are steps you can follow to solve the issue:

Check write permissions in the current directory: Make sure you have write permissions in the directory you're using. To do this, right-click on the directory, choose "Properties", then "Security", and check if your user account has write permissions. If you're unsure, try moving to a directory where you definitely have write permissions, like your user directory.

Check if there is enough disk space: You can check this by opening "This PC" (or "My Computer") and looking at the available space on your disk. If it's nearly full, you'll need to clear up some space.

Check the content of your listfile.txt: The file listfile.txt should contain the names of the .traineddata files (without the extension) you want to combine. Make sure these files exist in the current directory and your user has read permissions for these files.

Check your TEMP/TMP environment variable: Windows uses the TEMP or TMP environment variable to determine where to create temporary files. You can check this variable by opening a command prompt and typing echo %TEMP% or echo %TMP%. This will show the path where temporary files are created. Make sure this directory exists and that you have write permissions for it.

Run the command prompt as an administrator: Sometimes, running the command prompt as an administrator can solve the issue. To do this, search for "cmd" in the start menu, right-click on "Command Prompt", and select "Run as administrator". Then try the combine\_tessdata command again.

Update Tesseract: If none of the above works, consider updating your Tesseract software. It's possible that there's a bug in the version you're using. You can download the latest version from the official Tesseract GitHub page.

Remember, it's always important to run these commands in the command prompt and not in a Python or any other programming language shell.

If none of these solutions work, please provide any additional information or error messages you have.

User

in step 3, .lstmf file is listed rather than traineddata

ChatGPT

In Tesseract training, the .lstmf files are used in the training process to generate the .traineddata files. The combine\_tessdata command is used to extract, replace, or recombine components of a .traineddata file, not an .lstmf file.

If you're seeing .lstmf files listed in your listfile.txt, it seems like you might be confusing some steps of the Tesseract training process. If your goal is to train Tesseract with new data, you'll need to follow these general steps:

Prepare your training data: You will need to have images of the text you want to train on, and corresponding text files containing the correct text. This is often done by creating a .tif image file and a .txt file for each page of text.

Generate .lstmf files: Use Tesseract to generate .lstmf files from your images and text. This is done using the tesstrain.sh script provided by Tesseract. This script will generate a list of .lstmf files, which is usually called all-lstmf.

Train Tesseract: Use the lstmtraining command to train Tesseract. You will provide the list of .lstmf files as input, and this command will generate a .traineddata file as output. This involves several steps, including creating a model, training, and finally combining the trained data.

Test the trained data: You can then test the effectiveness of your trained data using the lstmeval command.

The combine\_tessdata command would only come into play if you already have a .traineddata file, and you want to modify it or extract information from it. It seems like you might be in the middle of the training process, so you might need to finish training before you can use combine\_tessdata.

User

it seems that I have just finished step 2, and the lstmf file path are listed in listfile.txt. Please help me to continue

ChatGPT

Great, you have done the groundwork. Now, let's proceed with the training process.

If you're done with step 2 and you have the .lstmf files listed in listfile.txt, then you are ready to start the actual training. Tesseract uses Long Short Term Memory (LSTM) networks, and the training for this has two stages:

Fine-Tuning or Transfer Learning: In this stage, you continue training from an existing trained model, such as the eng.traineddata provided by Tesseract.

Training from Scratch: It's only recommended for experts and it's not common to train from scratch.

For fine-tuning, which is most common, the general command is:

sh

Copy code

lstmtraining \

--model\_output /path/to/output/directory/my\_model \

--continue\_from /path/to/existing/model/eng.lstm \

--traineddata /path/to/existing/traineddata/eng.traineddata \

--train\_listfile /path/to/your/listfile.txt \

--max\_iterations 4000

model\_output is where the output files from the training will be written.

continue\_from is the path to the .lstm file of the model you're fine-tuning from. You can extract the .lstm file from a .traineddata file using the combine\_tessdata command like this: combine\_tessdata -e /path/to/existing/traineddata/eng.traineddata /path/to/output/directory/eng.lstm.

traineddata is the path to the .traineddata file of the language you're training.

train\_listfile is the path to the listfile.txt you created.

max\_iterations is the number of training iterations. The number 4000 is an example, you might need to adjust this based on your data.

Note: You need to replace the paths with your actual file paths.

You should periodically monitor the training process. The training can take a long time (several hours or even days) depending on the volume and complexity of your training data and the capabilities of your machine.

After the training process, lstmtraining will create checkpoint files in the output directory specified in the model\_output parameter, with names like my\_model\_checkpoint.

Once you decide that the training has proceeded enough (based on the reported accuracy), you can stop the training and use the lstmeval command to evaluate one of these checkpoints.

Then, you can use the lstmtraining command with the --stop\_training flag to convert the chosen checkpoint to a .traineddata file, which can be used for OCR:

sh

Copy code

lstmtraining --stop\_training \

--continue\_from /path/to/my\_model\_checkpoint \

--traineddata /path/to/existing/traineddata/eng.traineddata \

--model\_output /path/to/output/directory/eng.traineddata

The new eng.traineddata file created in your output directory is the final output of the training process, and can be used with the tesseract command for OCR.

Remember, all these commands need to be executed in the Command Prompt. Also, please ensure Tesseract and its training tools are correctly installed and their path is added to the system path.

User

Error, eng.lstm is an integer(fast) model, cannot continue training

ChatGPT

It seems like you're trying to continue training from an integer model, but Tesseract only supports continued training from floating point models. The default models provided by Tesseract are integer models due to their smaller size and faster operation.

To overcome this issue, you have two options:

Use a Floating-Point LSTM model: Tesseract provides some floating point models that you can use for continued training. These can be found in the tessdata\_best and tessdata\_fast repositories on the Tesseract GitHub page. For example, you might want to use the eng.lstm file from the tessdata\_fast repository. Once you've downloaded the .traineddata file, you can extract the LSTM file using the combine\_tessdata tool, like so:

bash

Copy code

combine\_tessdata -e eng.traineddata eng.lstm

Then, in the lstmtraining command, use eng.lstm as the input to the --continue\_from flag.

Start from Scratch: Although this isn't usually recommended because it requires a lot of data and can take a long time, you can start training a model from scratch. In this case, instead of the --continue\_from flag, you'll need to use the --net\_spec flag to specify the architecture of the network you want to train. Note that this will require quite a bit of expertise in LSTM networks to get good results.

I would suggest the first approach, using a floating point LSTM model for continued training. If you choose the second approach, please do sufficient research to understand the implications of the training parameters and network architecture.

User

deserialize header failed

ChatGPT

The error "deserialize header failed" is usually due to incompatibility between the LSTM model file and the Tesseract version you're using.

This could mean a few different things:

Corrupted or Incorrectly Formatted Model: If the LSTM model file you are trying to use is corrupt or not formatted correctly, Tesseract might fail to read the model and give this error. Try to download the model file again or check its integrity if possible.

Mismatch Between Tesseract Version and Model Version: The LSTM model file version needs to be compatible with the Tesseract version. For example, you may encounter this error if you're trying to use an LSTM model file that was trained with a newer version of Tesseract with an older version of Tesseract. You can try to update your Tesseract to the latest version to fix this issue.

Trying to Read Non-LSTM File as LSTM: If you accidentally use a non-LSTM file (like a .traineddata file) as an LSTM file, Tesseract would give this error. Make sure the LSTM file you're using is indeed an LSTM file, not a .traineddata file or any other file.

Using an Integer LSTM Model: Tesseract's training functionality only works with floating-point LSTM models, not integer LSTM models. If you're trying to use an integer LSTM model, you would get this error. The LSTM model files in the standard tessdata repository are integer models. For training, you should use a model from the tessdata\_best or tessdata\_fast repositories, which are floating-point.

Try to download the LSTM model again from the correct repository (tessdata\_best or tessdata\_fast), make sure you extract the LSTM file correctly from the .traineddata file, and try to use the LSTM file in the lstmtraining command again.

If none of these steps work, there could be an issue with your Tesseract installation or with your system. You might want to try reinstalling Tesseract, or try running the same commands on a different system or in a different environment to see if the error persists.

User

2 / 2

how to check step2

ChatGPT

To verify step 2, which is ensuring the LSTM file is correctly extracted and is the right type (floating-point LSTM model from tessdata\_best or tessdata\_fast repositories), you would need to:

Check the origin of your LSTM file: Ensure that your LSTM file is derived from the correct repository. Floating-point LSTM models can be found in the tessdata\_best and tessdata\_fast repositories on the Tesseract GitHub page. Make sure you have downloaded the .traineddata file from either of these repositories.

Check the extraction process: Once you have the right .traineddata file, you can extract the LSTM file using the combine\_tessdata tool, as shown below:

bash

Copy code

combine\_tessdata -e eng.traineddata eng.lstm

Ensure that the above command executes without any errors and that the eng.lstm file is created.

Check the LSTM file: Ensure that the LSTM file (e.g., eng.lstm) exists in your specified location. You can use the following command to check this:

bash

Copy code

dir eng.lstm

Or for Unix-based systems:

bash

Copy code

ls -l eng.lstm

If the file is there, it should show up in the output of this command.

Check the file size: The size of the LSTM file should be significant (several MBs), reflecting that it's a neural network model. A very small file size might indicate that the file is empty or corrupt.

Check the compatibility with your Tesseract version: If you're still getting an error, the issue might be with the version of Tesseract you're using. You can check your Tesseract version with the command tesseract --version. Generally, you should use the latest version of Tesseract, as this will have the most features and be the most compatible with the current LSTM models.

If after these checks you are still having issues, the problem might be with the specific LSTM model file you're using, or with your Tesseract installation. In that case, you might want to try using a different LSTM model, or reinstalling Tesseract.

User

for number ocr, which lstm model is recommended

ChatGPT

For OCR specifically focused on numbers, you could potentially still use the general English model (eng.traineddata), as it has been trained on a large variety of text, including numbers.

However, if you want to fine-tune a model specifically for digit recognition, you might want to consider training your own model with a dataset that consists mostly or entirely of numerical data. There is no pre-trained model specifically for numbers provided by the Tesseract project, as far as I know up to my knowledge cutoff in September 2021.

Here are the general steps you would follow to train your own model:

Collect a Dataset: This should be a collection of images containing the kind of text you want the model to recognize, along with ground truth labels. For a number recognition model, this could be a collection of images of numbers, taken from a variety of sources to ensure the model is robust to different fonts, sizes, and other variations. You might be able to find suitable datasets online, or you could create your own.

Generate LSTM Box Files: Use the Tesseract tesstrain.sh script to generate .lstmf files from your dataset. The tesstrain.sh script does this by running Tesseract on your images, generating box files that contain the bounding boxes of the recognized text, and then using these to generate .lstmf files.

Train the Model: Use the lstmtraining command to train the model, as I described in my previous messages. You can fine-tune from the existing eng.traineddata model or start training from scratch.

Remember that training a model is a complex task that can take a long time and require a lot of computational resources. If you're new to machine learning, I would recommend starting with the existing Tesseract models and only consider training your own model if the existing models don't meet your needs.

As always, be sure to check the latest Tesseract documentation for the most up-to-date information and detailed instructions on how to train a model.

User

how to fine-tune the existing

ChatGPT

Fine-tuning an existing Tesseract model, also known as Transfer Learning, involves training a new model starting from an existing trained model. This can be more efficient than training a model from scratch, especially if your new data is similar to the data the original model was trained on.

Here are the general steps you should follow:

Prepare your training data: Create a collection of images of the text you want the model to recognize, along with corresponding text files containing the correct text. For each page of text, you should have a .tif image file and a .gt.txt file with the same base name.

Generate .lstmf files: Use Tesseract to generate .lstmf files from your images and text. This can be done with the tesstrain.sh script provided by Tesseract, like this:

bash

Copy code

tesstrain.sh --fonts\_dir /path/to/your/fonts --fontlist 'Your Fonts' --lang eng --linedata\_only --langdata\_dir /path/to/langdata --tessdata\_dir /path/to/tessdata --save\_box\_tif --maxpages 10 --output\_dir /path/to/output

The tesstrain.sh script is a Linux script. If you're on Windows, you may need to adapt this process or use a Linux environment, such as the Windows Subsystem for Linux (WSL).

Extract the .lstm file from the existing traineddata file: You can do this using the combine\_tessdata tool provided by Tesseract, like this:

bash

Copy code

combine\_tessdata -e /path/to/existing/traineddata/eng.traineddata /path/to/output/eng.lstm

Train Tesseract with the new data: Use the lstmtraining command to continue training from the extracted LSTM file. The command looks like this:

bash

Copy code

lstmtraining --continue\_from /path/to/eng.lstm --model\_output /path/to/output/eng\_checkpoint --traineddata /path/to/existing/traineddata/eng.traineddata --train\_listfile /path/to/your/listfile.txt --max\_iterations 4000

The --train\_listfile parameter should point to a file containing a list of paths to your .lstmf files, one path per line.

Stop training and finalize the traineddata file: After training has proceeded for a while, you can stop the training and convert the most recent checkpoint into a .traineddata file that can be used with Tesseract for OCR:

bash

Copy code

lstmtraining --stop\_training --continue\_from /path/to/eng\_checkpoint --traineddata /path/to/existing/traineddata/eng.traineddata --model\_output /path/to/output/eng.traineddata

Remember to replace all the paths in the commands with the paths to your actual files.

This is a simplified overview, and the actual process may involve more steps and complexities, depending on your exact situation and requirements. Be sure to check the latest Tesseract documentation and training tutorials for the most up-to-date and detailed instructions.

User

what if to strart trainning from scratch

ChatGPT

Starting from scratch means training an entirely new model without basing it on an existing one. It requires a lot more data and time compared to fine-tuning. However, if you have a specific use case that's significantly different from any existing models, starting from scratch could be beneficial.

Here are the basic steps for training from scratch:

Prepare your training data: Similar to fine-tuning, prepare a collection of images of the text you want the model to recognize, along with corresponding text files containing the correct text. For each page of text, you should have a .tif image file and a .gt.txt file with the same base name.

Generate .lstmf files: Use Tesseract to generate .lstmf files from your images and text. This can be done with the tesstrain.sh script provided by Tesseract, as explained in the previous steps.

Create a list of .lstmf files: The .lstmf files created in the previous step should be listed in a text file (one per line). This file will be used for training. For example:

bash

Copy code

/path/to/page1.lstmf

/path/to/page2.lstmf

...

Train Tesseract with new data: Here's the command to start the training from scratch:

bash

Copy code

lstmtraining \

--model\_output /path/to/output/your\_model \

--net\_spec '[1,36,0,1 Ct3,3,16 Mp3,3 Lfys48 Lfx96 Lrx96 Lfx256 O1cNUM\_OF\_CLASSES]' \

--train\_listfile /path/to/your/listfile.txt \

--max\_iterations 4000

Replace NUM\_OF\_CLASSES with the number of different characters you want your model to recognize. This should include all alphanumeric characters (for English, that's 10 digits + 26 lowercase letters + 26 uppercase letters = 62) plus any special characters, and at least one more for handling 'blanks' or unrecognized characters. If you're doing digit recognition, NUM\_OF\_CLASSES would be 11 (10 digits + 1 blank).

your\_model is the basename of the output files. listfile.txt is the file that lists all your .lstmf files. 4000 is the number of iterations to run - you might need to adjust this based on your data and desired accuracy.

Finalize the traineddata file: After training has proceeded for a while, you can stop the training and convert the most recent checkpoint into a .traineddata file that can be used with Tesseract for OCR:

bash

Copy code

lstmtraining --stop\_training \

--continue\_from /path/to/your\_model\_checkpoint \

--model\_output /path/to/output/your\_model.traineddata

The net\_spec argument in the lstmtraining command defines the neural network specification, which consists of a series of layers with their types and parameters. You might need to adjust this specification based on your specific requirements and dataset.

Also, remember that training from scratch can be a lengthy process (it could take days or even weeks) and requires a substantial amount of labeled data.

User

must provide a -- traineddata see training documentation

ChatGPT

Apologies for any confusion. When training a model from scratch, you still need to provide a --traineddata file, which includes necessary information such as language-specific details and the list of characters that Tesseract should be able to recognize.

You should create a .traineddata file for your language if it doesn't already exist. This is typically done by using the tesstrain.sh script provided by Tesseract.

Here's how you might do it:

Create a .txt file with your training text: This text should include every character you want Tesseract to recognize. For OCR on numbers, this could just be "0123456789".

Run the tesstrain.sh script to generate .traineddata file:

bash

Copy code

tesstrain.sh --fonts\_dir /path/to/your/fonts --fontlist 'Your Fonts' --lang eng --linedata\_only --langdata\_dir /path/to/langdata --tessdata\_dir /path/to/tessdata --save\_box\_tif --maxpages 10 --output\_dir /path/to/output

Replace /path/to/your/fonts with the path to the directory containing the fonts you want to use for training, 'Your Fonts' with the list of fonts (as they are named in your system) you want to use for training, and /path/to/output with the directory where you want the .traineddata file to be saved.

Use the .traineddata file in the lstmtraining command:

bash

Copy code

lstmtraining \

--model\_output /path/to/output/your\_model \

--net\_spec '[1,36,0,1 Ct3,3,16 Mp3,3 Lfys48 Lfx96 Lrx96 Lfx256 O1cNUM\_OF\_CLASSES]' \

--traineddata /path/to/your/traineddata/eng.traineddata \

--train\_listfile /path/to/your/listfile.txt \

--max\_iterations 4000

Again, replace /path/to/your/traineddata/eng.traineddata with the path to your .traineddata file, /path/to/output/your\_model with the path and basename for the output files, /path/to/your/listfile.txt with the path to the list of your .lstmf files, and NUM\_OF\_CLASSES with the number of different characters your model should recognize.

You might need to adjust these steps based on your exact requirements, your dataset, and the latest Tesseract documentation and training tutorials.

User

lstmtraining --model\_output new\_model --net\_spec '[1,36,0,1 Ct3,3,16 Mp3,3 Lfys48 Lfx96 Lrx96 Lfx256 O1cNUM\_OF\_CLASSES]' --traineddata eng.traineddata --train\_listfile /path/to/your/listfile.txt --max\_iterations 4000 ; and it still reminds that must provide a --traineddata

ChatGPT

My apologies for the confusion, it appears there's a misunderstanding with the paths.

The --traineddata argument requires the full path to the .traineddata file, not just the file name. eng.traineddata is the filename, but the system does not know where this file is located. You need to provide the full path to the file.

For example, if eng.traineddata is located in the directory /home/user/tessdata, then you would use the following command:

bash

Copy code

lstmtraining \

--model\_output /path/to/output/new\_model \

--net\_spec '[1,36,0,1 Ct3,3,16 Mp3,3 Lfys48 Lfx96 Lrx96 Lfx256 O1cNUM\_OF\_CLASSES]' \

--traineddata /home/user/tessdata/eng.traineddata \

--train\_listfile /path/to/your/listfile.txt \

--max\_iterations 4000

Replace /path/to/output/new\_model with the path and basename for the output files, /path/to/your/listfile.txt with the path to the list of your .lstmf files, and NUM\_OF\_CLASSES with the number of different characters your model should recognize.

If you're using Windows, the path will look a bit different, like C:\path\to\tessdata\eng.traineddata. You can determine the path by finding the file in your file explorer, right clicking, and selecting 'Properties'. The path should be listed there.

Also, remember that you can't use the default eng.traineddata file provided with Tesseract if you are training from scratch and your language or character set is not English. In that case, you need to create a new .traineddata file for your language or character set, as described in the previous steps.

User

no use ,strill reminds that must provide a --traineddata

ChatGPT

I apologize for the confusion. It's possible that Tesseract is not able to find your eng.traineddata file at the path you specified.

Let's try the following steps:

Check the traineddata file: Please confirm that eng.traineddata exists in your specified directory. You can do this by navigating to the directory and ensuring that the eng.traineddata file is there.

Check the path: Please ensure that you are providing the correct path to your eng.traineddata file in the command. The path should be the absolute path (starting from the root of your filesystem) to the eng.traineddata file.

For example, if your eng.traineddata is in the tessdata directory within your home directory, the path would be something like /home/username/tessdata/eng.traineddata on Linux or C:\Users\username\tessdata\eng.traineddata on Windows.

Check your Tesseract version: Ensure that your Tesseract version is 4.0 or above, because LSTM training is not available in earlier versions.

Run the command again: Try running the lstmtraining command again with the correct --traineddata argument. Make sure you include the full path to the eng.traineddata file.

If these steps do not resolve your issue, there might be something else going on. In that case, please provide the complete error message or any other information about your setup, so I can better understand the issue and help you resolve it.

User

if in the continue\_from mode, how to evaluate the lstmf file is correct

ChatGPT

The LSTM files (.lstmf) used for training contain training samples that include both the input image and the expected output text.

If you want to validate these files, there are a couple of ways you could do it:

Use Tesseract to read the files: You can use Tesseract's lstmeval command to evaluate the LSTM files against a trained model. The command would look something like this:

bash

Copy code

lstmeval \

--model /path/to/your/model.traineddata \

--traineddata /path/to/tessdata/eng.traineddata \

--eval\_listfile /path/to/your/listfile.txt

This command will output evaluation metrics like Character Error Rate (CER) and Word Error Rate (WER), which can give you an idea of how well the trained model is performing on the data in the LSTM files.

Replace /path/to/your/model.traineddata with the path to the trained model you want to evaluate, /path/to/tessdata/eng.traineddata with the path to the eng.traineddata file, and /path/to/your/listfile.txt with the path to the list of your .lstmf files.

Inspect the files manually: You can use Tesseract's lstmbox command to generate box files from LSTM files. The command would look like this:

bash

Copy code

lstmbox --input /path/to/your/file.lstmf

This will output box data to the console. Each line of box data corresponds to one character in the image, with the character, the position of the bounding box, and the page number. You can check if the characters and their positions match the images.

Replace /path/to/your/file.lstmf with the path to the .lstmf file you want to inspect.

Remember to replace all the paths in the commands with the paths to your actual files. Also, these are simplified examples, and the actual process may involve more steps and complexities, depending on your exact situation and requirements.

User

there seems some problem in lstmf,. First document cannot be empyt! and num\_pages\_per\_doc\_>0

ChatGPT

It seems like Tesseract is unable to find any valid data in the .lstmf files listed in your listfile.txt. The error messages "First document cannot be empty!" and "num\_pages\_per\_doc\_ > 0" suggest that the LSTM file is either empty or doesn't contain valid training data.

Here are a few things you could do to resolve this:

Check your .lstmf files: Make sure your .lstmf files are not empty and do contain valid training data. You can inspect them manually using the lstmbox command I mentioned in the previous response.

Check your listfile.txt: Make sure the paths to the .lstmf files listed in your listfile.txt are correct. Each line of listfile.txt should contain the full path to one .lstmf file.

Check the process of generating .lstmf files: The .lstmf files should be generated using the tesstrain.sh script provided by Tesseract. This script takes as input .tif image files and corresponding .gt.txt text files containing the correct text. If there is an issue with either the images or the text files, the generated .lstmf files may not contain valid training data.

If you've confirmed all of the above and are still experiencing issues, there may be a problem with your Tesseract installation or the version of Tesseract you're using. You might need to reinstall Tesseract or update to the latest version.

Please note that these are general suggestions based on the information available, and the actual solution may depend on your specific situation and setup. If you continue to experience issues, it might be helpful to provide additional information or seek help from the Tesseract community or support resources.

User

lstmbox 不是内部或外部命令

ChatGPT

The error message "lstmbox 不是内部或外部命令" is in Chinese and translates to "lstmbox is not an internal or external command". This suggests that the command is not recognized by your system.

Please note that as of my knowledge cutoff in September 2021, there is no lstmbox utility provided by Tesseract.

If you want to visually inspect the boxes in your training data, you can use the tesseract command to output a .box file and then use a box file viewer to visualize the boxes. Here is how you can generate a .box file:

bash

Copy code

tesseract imagename.tif imagename batch.nochop makebox

This will create a imagename.box file with the recognized text and box data.

You can then use a box file viewer (there are several available online) to open the image and the box file together. This will overlay the boxes on the image, so you can see if the boxes correctly correspond to the characters.

However, this will only check the bounding boxes, not the LSTM features stored in the .lstmf files. To inspect the LSTM features, you would need to use the Tesseract API in a programming language like Python or C++ to read the .lstmf files and interpret the data. This could be complex and is usually not necessary for normal training tasks. If you suspect there is an issue with the .lstmf files, it might be better to try regenerating them or checking the process you used to generate them.

Please note that all information is accurate as of my knowledge cutoff in September 2021, and the available commands or utilities might have changed if you are using a later version of Tesseract. If you continue to experience issues, it might be helpful to provide additional information or seek help from the Tesseract community or support resources.

User

tesseract C:\Users\ZJLZ1026.LZSJY\PycharmProjects\ocr\_test\ocr\_train2\0\6-26-11-1-4-5-region\_2-crop.tif C:\Users\ZJLZ1026.LZSJY\PycharmProjects\ocr\_test\ocr\_train2\0\6-26-11-1-4-5-region\_2-crop batch.nochop makebox

Empty page!!

Empty page!!

ChatGPT

2 / 2

The "Empty page!!" message means that Tesseract was unable to find any text regions in the image you provided.

There are a few possible reasons for this:

The image is actually empty: It's possible that the image doesn't contain any text or the text is too small or faint to be recognized by Tesseract. You should open the image and check if it contains visible text.

The image is in an unsupported format: Tesseract supports a variety of image formats including TIFF, PNG, JPEG, and others. However, some formats or sub-formats might not be supported. If your image is not in one of the common formats, you might need to convert it to a supported format.

There's an issue with the image file: Sometimes, image files can be corrupted or have other issues that prevent Tesseract from reading them correctly. You could try opening the image in an image viewer or editor to see if there are any obvious issues.

Tesseract's page segmentation mode is not appropriate for the image: Tesseract uses a method called page segmentation to find and recognize text regions in an image. By default, Tesseract assumes that the image contains a complex layout with multiple text regions. If your image contains only a single block of text, or if the layout is otherwise unusual, Tesseract might not be able to find the text. You can try setting a different page segmentation mode using the -psm option. For example, to set the page segmentation mode to 'single block', you can use -psm 6.

If none of these suggestions resolve the issue, there might be something else going on. In that case, please provide any additional information you have, and I'll do my best to help you troubleshoot the problem.

User

1. image could be open correctly ;2. tif file is saved;3. only one char(number) in each tif file

ChatGPT

Given your specific scenario where each .tif image contains only a single character, the issue could be due to the Page Segmentation Mode (PSM) Tesseract is using to process the images. By default, Tesseract is optimized for recognizing multiple lines of text, which may result in failure when the image contains only a single character.

You can adjust the PSM by using the -c tessedit\_pageseg\_mode=6 or -c tessedit\_pageseg\_mode=10 options in your command. Mode 6 assumes a single uniform block of text, and mode 10 treats the image as a single character.

Here is your updated command:

bash

Copy code

tesseract C:\Users\ZJLZ1026.LZSJY\PycharmProjects\ocr\_test\ocr\_train2\0\6-26-11-1-4-5-region\_2-crop.tif C:\Users\ZJLZ1026.LZSJY\PycharmProjects\ocr\_test\ocr\_train2\0\6-26-11-1-4-5-region\_2-crop -c tessedit\_pageseg\_mode=10 batch.nochop makebox

This should help Tesseract recognize the single character in your image. If you still encounter issues, please let me know so we can continue troubleshooting.

User

how to evaluate if the lstmf file is correct? and the tif file is 4kb, what is the lstmf file size is estimated?

ChatGPT

Evaluating the correctness of an .lstmf file can be a bit tricky because they are binary files containing serialized LSTM features. These files are not intended to be read or interpreted directly by humans.

The lstmtraining and lstmeval tools provided by Tesseract can read .lstmf files and use them for training and evaluation. If an .lstmf file is valid, lstmtraining should be able to use it without errors.

However, if you need to inspect the files more closely, you would likely need to write a script using Tesseract's API in a language like Python or C++. This could be a complex task and is usually not necessary for normal training tasks. If you suspect there is an issue with the .lstmf files, it might be better to try regenerating them or checking the process you used to generate them.

Regarding the size of the .lstmf files, it can vary significantly depending on the size and complexity of the images and the amount of text they contain. A .lstmf file could be larger or smaller than the original image file, but it's hard to estimate the size without knowing more about the specifics of the image and text.

It's important to note that the size of an .lstmf file by itself does not necessarily indicate whether the file is valid or not. However, if an .lstmf file is significantly smaller than expected or if its size is 0, that could be an indication of an issue with the file.

If you have any other questions or need further clarification, please let me know!

User

Finished! Selected model with minimal training error rate (BCER) = 0, what is the next step to achieve .trainneddata

ChatGPT

Great job! You've finished the training phase. Now that you've got a trained model with a satisfying error rate, you can move on to the next step, which is to create the traineddata file for Tesseract.

Here's the process:

Combine the network specification file with the trained model:

bash

Copy code

lstmtraining --stop\_training \

--continue\_from /path/to/your/output\_checkpoint \

--traineddata /path/to/your/eng.traineddata \

--model\_output /path/to/output/location/final.traineddata

This command tells Tesseract to stop training and combine the trained model with the network specification to produce a traineddata file. Replace the paths with the correct ones for your setup.

Test the traineddata file:

Now that you've got your traineddata file, you should test it to ensure it's working as expected. You can do this using the Tesseract command, similar to this:

bash

Copy code

tesseract /path/to/test/image.tif stdout -l eng --oem 1

Replace /path/to/test/image.tif with the path to an image file you want to use for testing. This should output the recognized text to the console.

Use the traineddata file:

Now you're ready to use your traineddata file! You can use it just like you would use the default Tesseract traineddata files. Simply specify your traineddata file using the -l option when you run the Tesseract command.

Please replace all paths in the commands with the paths to your actual files. If you encounter any issues or have any questions, please let me know!

User

Onemore thing, is that possible to simplify/ reduce the traineddata model, by limiting the char of number

ChatGPT

Yes, if you're training a model to recognize a limited set of characters, you can create a new training set that only includes those characters, which may simplify the model and improve performance for your specific use case.

Here's how you can do it:

Create a new unicharset file: The unicharset file is a file that tells Tesseract which characters (unichars) it should recognize. You can create a new unicharset file that includes only the characters you want. This can be done manually or by using the unicharset\_extractor tool on a set of box files that include only your desired characters.

Create new box and .lstmf files: Using your training images, create new box files that include only your desired characters. Then, use these box files to create new .lstmf files.

Train a new model: Using your new .lstmf files and the new unicharset, train a new model as you did before. The resulting model should be simpler and possibly smaller than the original model.

However, please be aware that by reducing the set of recognized characters, you're also reducing the model's ability to correctly recognize other characters. If the model encounters a character that's not included in your limited set, it won't be able to recognize it correctly. This may be fine if you're certain that the model will only ever need to recognize your limited set of characters, but it could cause issues if that's not the case.

Lastly, it's important to note that while this process can potentially simplify the model and improve performance for recognizing your specific set of characters, it's not guaranteed to do so. The effectiveness of this approach can depend on a variety of factors, including the quality and variety of your training data and the specifics of your use case.

User

please provide more detail on step 1, about to create unicharset\_extractor

ChatGPT

Certainly! unicharset\_extractor is a utility provided by Tesseract to generate a unicharset file, which essentially describes all the unique characters (unichars) present in the training data.

Given you want to limit the characters to numbers only, the first step would be to create .box files that contain only the characters you're interested in. Since you're focusing on number OCR, the .box files should only contain boxes for digits 0-9.

Once you have prepared such .box files, you can use unicharset\_extractor as follows:

bash

Copy code

unicharset\_extractor /path/to/your/boxfiles/\*.box

This command will create a file called unicharset in the current working directory. This file will contain all the unique characters found in the provided box files.

Replace /path/to/your/boxfiles/\*.box with the actual path to your box files. The \*.box part of the command tells the unicharset\_extractor to process all files in the specified directory that have the .box extension.

After running this command, you should inspect the generated unicharset file to make sure it only contains the characters you want (in your case, digits 0-9). If you find unwanted characters, you should go back to your box files and correct any errors. Once your unicharset file is correct, you can proceed with the other training steps.

Remember that you should use the same unicharset file in all subsequent steps of the training process. This ensures that Tesseract knows exactly which characters it should be trying to recognize.

If you have any more questions or need further clarification, please let me know!