OCROPY

Text Recognition Software

Umesh Rao

Completed in partial requirements of LTEC 647C

The Department of Education

University of Hawaii

Table of Contents

Illustration Index

1 ABOUT AND ACKNOWLEDGMENTS

I created this document as project work for LTEC 647C, a graduate course of Learning Design and Technology program, Department of Education, University of Hawaii. LTEC 647 C is a course about “`Exploration and evaluation of new tools and strategies for teaching and learning. (C) free/open software;`”.

I am also working part time, in the digitization of old text for the Digital Arts and Humanities Initiative, Department of Arts and Humanities, University of Hawaii. I use Ocropy for this work. I thank the director of this initiative, Dr. Richard Rath, for having provided me this opportunity. Do check out their cool new website, http://dahi.manoa.hawaii.edu/.

The document is not about algorithms employed in Ocropy. The algorithms have changed over time. In fact Thomas Breuel’s original journal paper on Ocropus is outdated (Thomas M. Breuel 2008).

The document is not a tutorial. This would require, ideally, packaging it with an example image and maybe the results after every processing step as well.

Ocropy is a collection of programs, rather than one software with one interface. There are other frameworks based off of Ocropus, such as Ocrocis. The document does not describe these frameworks.

The document is not a complete training module as well. This would require both an explanation of the algorithms used by the programs and implement an instruction module, both not provided here.

The morphing of the document was to bring value add to its users. The document does touch upon the important aspects of the programs and provide tips on how to get better performance. Some of them are from my experience and some are from the experience of other users. For example, Ocropy makes mistakes recognizing capital letters when combined with normal letters. This information is listed in one of the issues on Ocropy’s website. It is information that you need to get off and running with Ocropy, without having to spend weeks “tuning” it. It is information that DAHI members could use.

2 INTRODUCTION

Ocropus is a open source document analysis and character recognition software (OCRopus 2017). It was originally developed by Thomas Breuel and is intended for large-scale digital library applications (Thomas Breuel 2008). It is a collection of programs for document analysis that may require some prior image processing and train new models tailored to a document (Tom 2017b). Beyond input line normalization, all that is required for Ocropus is choosing the input size (current default is 32) and the number of internal state units (currently 100). This assumes the input image is at least 300 dpi (dots per inch) resolution.

```
“As an indication: An A4-paper (210mm × 297mm) scanned with 300dpi results in an image with
2504 x 3540 pixels.”
```

It was originally written in C++ and now coded in python. The python version is called Ocropy. There is also a C++ version called clstm (Tom 2017a). Ocropus is released under the Apache Version 2 license.

Ocropus:

- does not segment the lines into individual characters, instead divides the text lines into regions and uses shape models to make the detection of these regions more robust (Breuel-LSTM-OCR-ICDAR13.Pdf n.d.)
- is context aware processing and recognize patterns occurring in time series, based on Long Short Term Memory neural networks
- uses a normalization process to scale the handwritten text lines with varying writer speed and non-uniform skew

Steps for recognition of text using Ocropus are:

- Generating **ground truth**, text that matches the text in the image
- **Train**ing the Ocropus neural network with the ground truth generated from previous step
- **Evaluate** the effectiveness of the model so generated on new text images
- Apply the most effective model so evaluated, for text **recognition** of in other images, typically from the same book

Block Diagram

A block diagram of the Ocropus workflow is shown in the two figures below (Thomas M. Breuel 2008) (Workflow.Vue n.d.). Roughly it is

```
    Ocropus-nlbin → Ocropus-gpageseg → (Ocropus-rtrain) → Ocropus-rpred →..
    .. Ocropus-hocr → Ocropus-visualize-results → Ocropus-gtedit.
```

An input image or images in the form of a book and a Ocropus model file are the basic inputs.

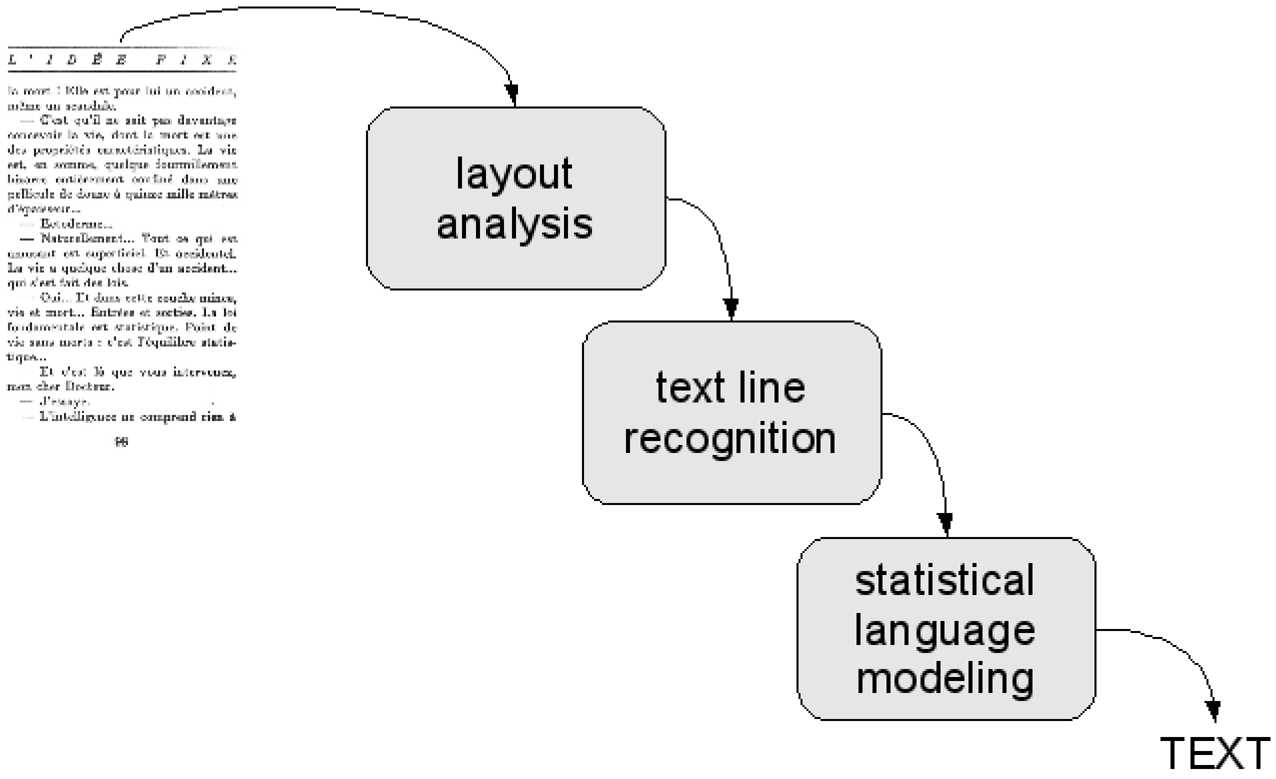


Illustration 1: Block Diagram of Ocropy

The more detailed block diagram below shows “input font” as input to ocropus-linegen. This is somewhat misleading. This does not mean that Ocropus needs to be trained using individual letters of a font, as is done with other similar tools. Ocropus-linegen is used to generate images for training where the text is known and the font has been chosen. Typically the input image is / are generated by scanning a page / pages in a document and the text font is not known.

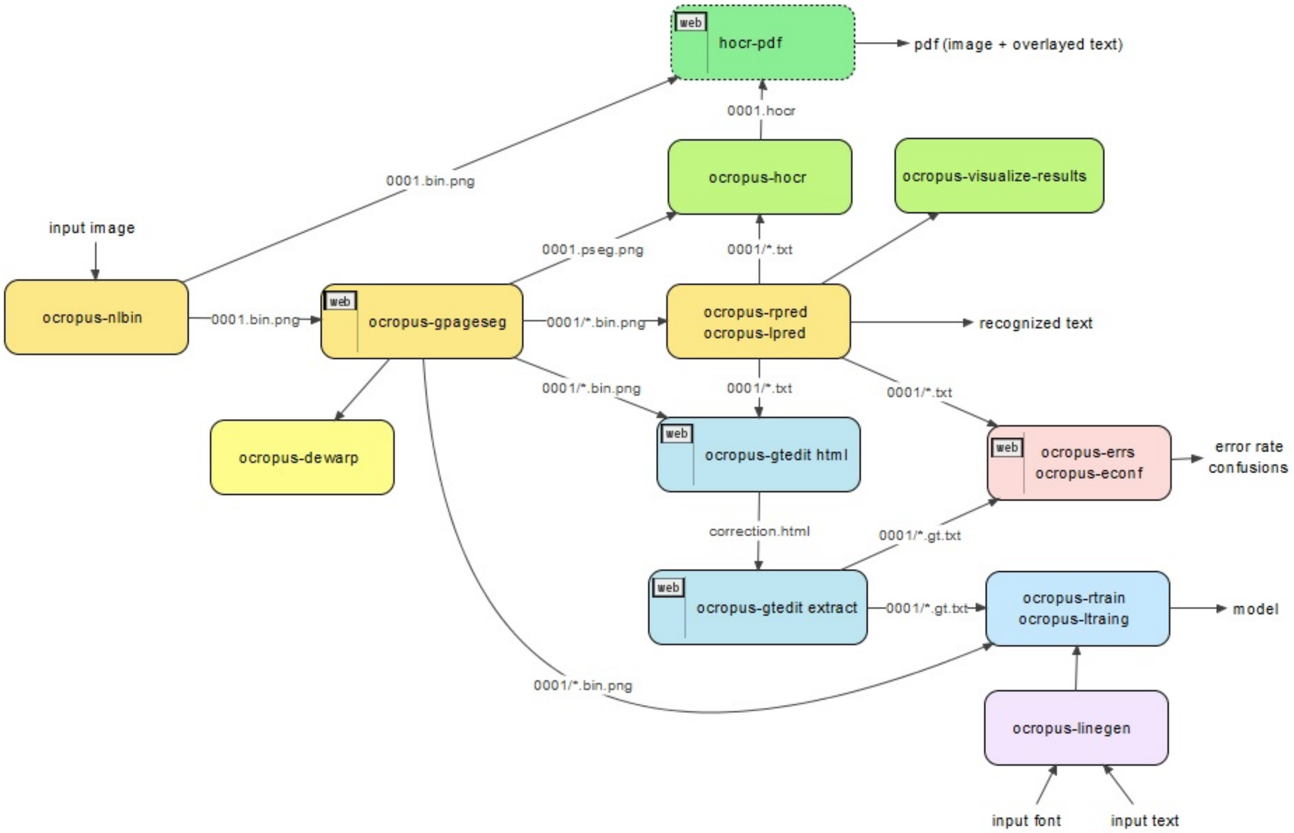
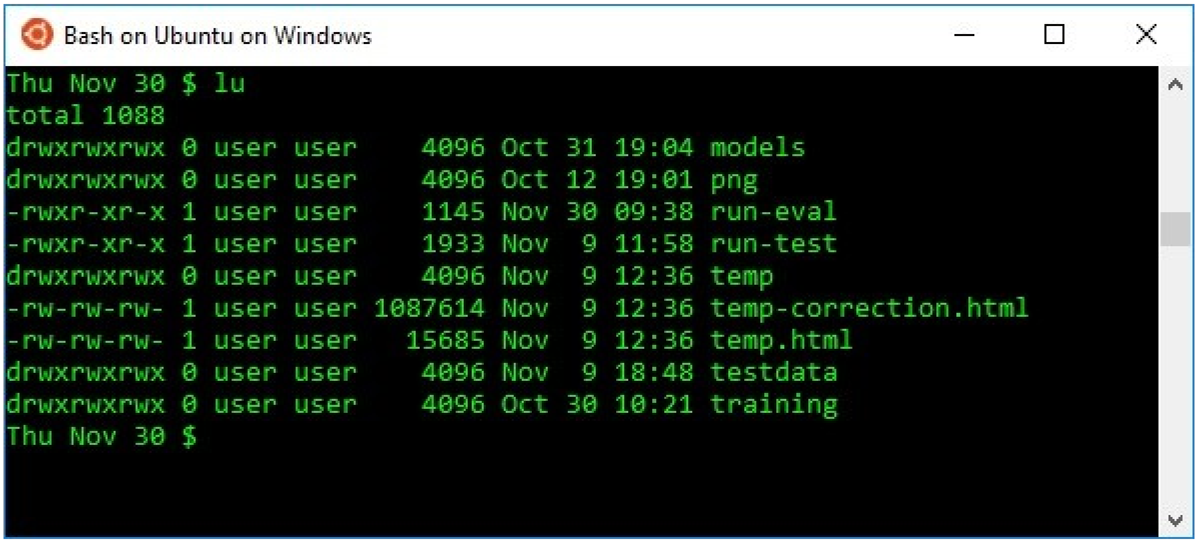


Illustration 2: Detailed block diagram of work flow in Ocropy

Directory Structure

It is a good idea to keep the training and evaluation data in two directories (Practical Experience with OCRopus Model Training : Arbeitsgruppe Kommunikationssysteme n.d.). The directory list looks like as shown in the figure below.



```
Thu Nov 30 $ lu
total 1088
drwxrwxrwx 0 user user    4096 Oct 31 19:04 models
drwxrwxrwx 0 user user    4096 Oct 12 19:01 png
-rwxr-xr-x 1 user user    1145 Nov 30 09:38 run-eval
-rwxr-xr-x 1 user user    1933 Nov  9 11:58 run-test
drwxrwxrwx 0 user user    4096 Nov  9 12:36 temp
-rw-rw-rw- 1 user user 1087614 Nov  9 12:36 temp-correction.html
-rw-rw-rw- 1 user user   15685 Nov  9 12:36 temp.html
drwxrwxrwx 0 user user    4096 Nov  9 18:48 testdata
drwxrwxrwx 0 user user    4096 Oct 30 10:21 training
Thu Nov 30 $
```

Illustration 3: Typical Ocropy directory list

Ocropy writes its output files by default, in a directory called temp. Ocropy creates a directory within temp for each image that it processes and names them as numbers 0001, 0002, etc.

```
temp/
        0001/
                010001.bin.png
                010001.txt
                010002.bin.png
                010002.txt
...
        0002/
…
```

Ocropy creates an image file and a text file for each line it encounters in an image and once again names them as numbers 010001, 010002, .. 010009, 01000a, .. 01000f, 010010, etc. Each line recognized causes two files to be output, a .bin.png image file and a .txt file that contains the recognized text. Some info on the directories seen here:

- png directory contains input images in .png format
- models directory contains models in .gz format, created during training
- testdata directory contains .bin.png and .gt.txt ground truth files for evaluating a model file
- run-test.sh is script to run Ocropy, an example is listed in Appendix A
- run-eval.sh is a script to evaluate the effectiveness of an Ocropy model, an example is listed in Appendix B.

3 GENERATING GROUND TRUTH

Ocropus-rtrain is used to produce the model file. Training requires images of single line text and their corresponding ground truth text (Training an Ocropus OCR Model n.d.). It expects the truth data to be in the .gt.txt files with the same name as the PNG files for the names.
For example,

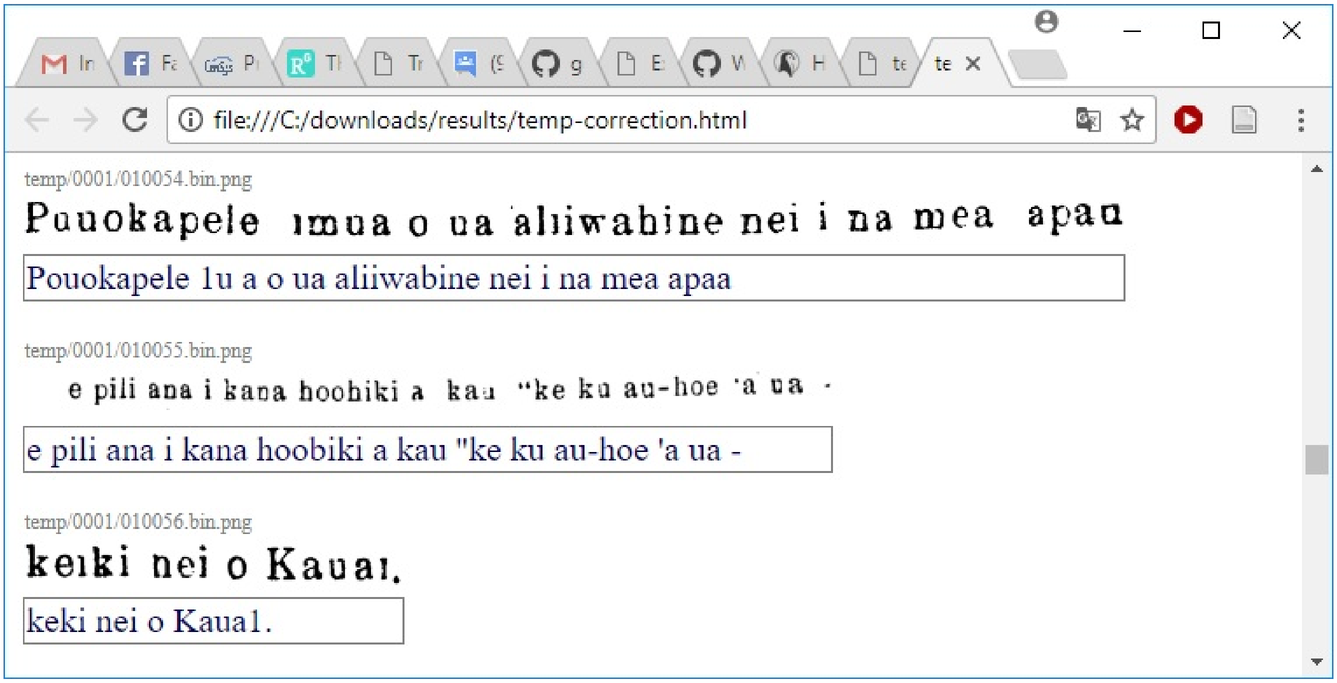
```
book/0001/010001.png
book/0001/010001.gt.txt
```

Each .png and the corresponding .gt.txt file may be considered a vector of the training set.

The ground truth text file is produced in two ways. In one, you use the default model file supplied by Ocropy, en-default.pyrnn.gz, to run the entire flow and produce recognition results in the form of a html file, temp-correction.html.

```
    Ocropus-nlbin → Ocropus-gpageseg → Ocropus-rpred →..
    .. Ocropus-hocr → Ocropus-visualize-results → Ocropus-gtedit → temp-correction.html
```

You then edit temp-correction.html with a web browser such as firefox and correct the errors.



file:///C:/downloads/results/temp-correction.html

temp/0001/010054.bin.png

Pouokapele 1u a o ua aliiwabine nei i na mea apaa

temp/0001/010055.bin.png

e pili ana i kana hoobiki a kau "ke ku au-hoe 'a ua -

temp/0001/010056.bin.png

keki nei o Kaua1.

Illustration 4: temp-correction html file for producing ground truth

Illustration 3 shows part of an example temp-correction.html file. You can see the lines vary in letter size, skew and font. Vector 010056 editable text box reads “keki nei o Kaua1” and needs to be corrected as “keiki nei o Kauai”. Once edits are done, the file is saved and the ground truth extracted by running ocropus-gtedit again:

```
> ocropus-gtedit extract temp-correction.html
```

The .gt.txt files will be written into temp directory, in this case, into temp/0001. In doing this, all the lines from temp-correction.html file are converted to vectors, each represented exactly once.

Alternately you could manually edit the bin.png and .txt files produced by the predictor, Ocropus-rpred, to create the training set:

```
   Ocropus-nlbin → Ocropus-gpageseg → Ocropus-rpred → ??????.bin.png and ??????.txt files
```

The predictor uses the default model, en-default.pyrnn.gz, for predicting the text. To create the ground truth, rename all *.txt files as *.gt.txt files and edit them individually, so that they match what is in the corresponding .png file. You can then choose your vectors for the ground truth and replicate a vector multiple times if you so wish. More about this in the training section.

4 TRAINING

Ocropus trains a model by learning from the differences between its predictions and the ground truth. As it does the compare, it adjusts the weights of the neural net to get the results of the neural net closer to the ground truth text. For the same reason, a methodology that works is to use the default model file to begin with and utilize the lines that cause errors as elements of the training set.

The default model has not seen typewriter fonts nor has it seen ALLCAPS text (Training an Ocropus OCR Model n.d.). Choose book pages or images that contain ALLCAPS letters as they tend to occur less frequently. The lines so generated, split them into training and evaluation data, the majority of it going for training.

Look for the frequency of occurrence of the letters / words in the training set and deliberately repeat vectors that will increase the frequency of the ones that occur less (Practical Experience with OCRopus Model Training : Arbeitsgruppe Kommunikationssysteme n.d.).
“To check which characters occur in you training data you can use this command:

```
  cat training/*.gt.txt | sed 's/\(.\)/\1\n/g'| sort | uniq -c
```

and your evaluation data as well:

```
  cat evaluation/*.gt.txt | sed 's/\(.\)/\1\n/g'| sort | uniq -c
```

“

Ocropus trains one line at a time, adjusting its weights to reduce the differences. “It transcribes the text in a line, then adjusts the weights in the Neural Net to compensate for the errors. Then it does this again for the next line, and the next, and so on. When it gets to the last line of labeled data, it starts over again. As it loops through the training data over and over again, the model gets better and better.” (Training an Ocropus OCR Model n.d.). “This produces lots of output like this:

```
2000 70.56 (1190, 48) 715641b-crop-010002.png
   TRU: u'504-508 West 142nd Street, adjoining and west of Hamilton'
   ALN: u'504-5088 West 422nd Street, adjoining and west of Hammilton'
   OUT: u'3od-iS est 4nd Street, doning nd est of Sarilton'
2001 32.38 (341, 48) 726826b-crop-010003.png
   TRU: u'NO REPRODUCTIONS'
   ALN: u'NO REPRODUCTIONS'
   OUT: u'sO EROCoOri'
...
```

TRU is the truth data. OUT is the output of the model. ALN is a variant of the model output which is aligned to the truth data. It's used to adjust the model weights more precisely. It typically looks better than the model output, especially in early iterations. It lets you know that you're making progress.”

Since the default model has unicode characters, it is better to set the environment variable,
> export PYTHONIOENCODING=”UTF-8”

The character set of the default model cannot be changed. When the images have special characters that are not in the default model, you have to use the -c or -codec option.

```
ocropus-rtrain -o model *.png -c *.gt.txt > result.log 2>&1
```

The default model has 157 characters that Ocropus trained with. The log file shows this.

```
# using default codec
# charset size 157 [ ~!"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}¡¢
£§©«®°¶»¿ÀÂÄÆÇÈÉÊËÎÏÔÖÙÛÜßàâäæçèéêëîïôö÷ùûüÿŒœŸ†‡•‣‹›€·▪▫◦]
```

One disadvantage of not using the default model during the training is that now you have to train Ocropus on all the alphanumeric characters and the symbols as well. If your training set did not contain any of these letters, it will cause an error if such a letter occurred during recognition stage. There is an ongoing pull request to expand or even contract the codec, along with the corresponding LSTM layer (Implementation of Resizing Codec by ChWick · Pull Request #277 · Tmbdev/Ocropy n.d.).

5 EVALUATION

Fix typos in the ground truth. Run a prediction on the training data and check differences between ground truth and the prediction. If using the generated model for prediction causes the errors to be consistently high and in double digits, it is an indication there are typos in the ground truth.

Ocropy supplies python scripts to evaluate the model. A bash script that uses this is shown in Appendix B. Ocropus-errs finds the differences between the prediction and the ground truth, while ocropus-econf calculates a distance between them. The script will run the evaluation for every model file it finds in the specified directory.

Output from Ocropus-errs looks like this:

```
errors 9 missing 0 total 856 err 1.051 % errnomiss 1.051 % 0.0105140186916
```

The run-eval script extracts the errors for each such model in the errs.txt file.

```
135000            9     1.051
136000            5     0.584
137000            7     0.818
138000           10     1.168
139000            4     0.467
```

The first column is the numbers extracted from the model file name and so shows the number of iterations at which the model was generated. The second column shows the number of errors and the third column shows the percentage of errors. Typically the number of errors will reach a plateau and then rise again. You can then choose the model file where the errors reached a plateau.

6 IMAGE PREPROCESSING

To get to the text in the image, it has to be prepared for better performance of the subsequent steps. This is true at every step. Binarization is a step where the contrast between the information and the background is enhanced. It is sort of like a simple noise reduction system. Binarization is when all pixels round off to either 0 or maximum value. In its simplest form, a threshold is used to determine if the pixel is 0 or maximum value. In Ocropy, this is handled by ocropus-nlbin.

Then there is noise reduction and skew correction before image segmentation. These are also handled by ocropus-nlbin. It is important for the binarized image to not exhibit any skew. Open the .bin.png image in temp directory and look at the edges of the columns. If they show an angle towards or away from a corner, try increasing the skewsteps parameter to ocropus-nlbin. The more the skewsteps the more time it takes to calculate the skew and the more accurate is its calculation and less the angle.

The more accurate the skew calculation, the better the segmentation results will be. If the image contains multiple columns, such as for a newspaper, you could manually split the image into its individual columns. You can then compare the skew Ocropus calculates for the individual columns against what it calculates for the complete page.

The image is then segmented to take into account paragraphs, columns and non-text matter, this is handled by ocropus-gpageseg. You can find more information about the segmentation algorithms used in Ocropy, in Amy's thesis work (Winder, Andersen, and Barney Smith 2011). A column separator could be white space or a black vertical line.

The two most important parameters of ocropus-gpageseg are max column separators and min height. If white space between words get aligned in consecutive lines, that white space may start to look like a column separator. The min height prevents this. Tables in the document can cause more column separators than usual. You could increase the max column separators argument to take this into account. The wiki lists these arguments (Tmbdev/Ocropy n.d.).

```
  --maxcolseps MAXCOLSEPS
                        maximum # whitespace column separators, default: 3
  --csminheight CSMINHEIGHT
                        minimum column height (units=scale), default: 10
```

When you have tables in the page, chances are the columns are black lines. The black column separator is used for this.

```
  --maxseps MAXSEPS     maximum black column separators, default: 2
  -b, --blackseps       also check for black column separators
```

Currently the default min height for black column separator is about 20 and is not parameterized.

7 RECOGNITION

Recognition is the step that produces the text from the image. This is handled by Ocropus-rpred. This is the one that reads the model file that is produced during training. There is a page on this in the Ocropy wiki. Any time a line does not have any characters or no characters is recognized, there is an error produced,

```
File "/usr/local/lib/python2.7/dist-packages/ocrolib/common.py", line 107, in write_text
if not nonl and text[-1]!='\n':
IndexError: string index out of range
```

Another common occurrence is where the images have special characters and –codec option. There is a python file that declares the character set Ocropus has been trained for, called chars.py. On my machine, it resides in /usr/local/lib,

```
/usr/local/lib/python2.7/dist-packages/ocrolib/chars.py
```

When there are special characters that are not available in the default character set and you use the -c option during training, edit the chars.py file to include those additional characters in your character set. The character set defined in chars.py has to reflect the character set that Ocropus (model) is trained for.

```
digits = u"0123456789"
letters = u"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
symbols = ur"""!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~"""
ascii = digits+letters+symbols

xsymbols = u"""€¢£»«›‹÷©®†‡°·•◦‣¶§÷¡¿▪▫"""
german = u"ÄäÖöÜüß"
french = u"ÀàÂâÆæÇçÉéÈèÊêËëÎîÏïÔôŒœÙùÛûÜüŸÿ"
turkish = u"ĞğŞşıſ"
greek = u"ΑαΒβΓγΔδΕεΖζΗηΘθΙιΚκΛλΜμΝνΞξΟοΠπΡρΣσςΤτΥυΦφΧχΨψΩω"
mychars = u"˓ʻˊăúáó"

#default = ascii+xsymbols+german+french
default = ascii+xsymbols+german+french+mychars
```

Just an aside. An interesting thing happens when you do train with the default model and you do have characters that are special. Ocropus dutifully trains on all those special characters and reports good output.

```
The training output shows that the output of the RNN is correct,
119004 0.18 (530, 48) 010010.bin.png
TRU: u'12. Hi\u02bbo\u02ca kini akua o Ka-hiki;'
ALN: u'12. Hi\u02bbo\u02ca kini akua o Ka-hiki;'
OUT: u'12. Hi\u02bbo\u02ca kini akua o Ka-hiki;'
```

Even if you did declare them in chars.py, during recognition Ocropus assigns some other characters in place of each of the special characters, although it is consistent with the assignment. For example,

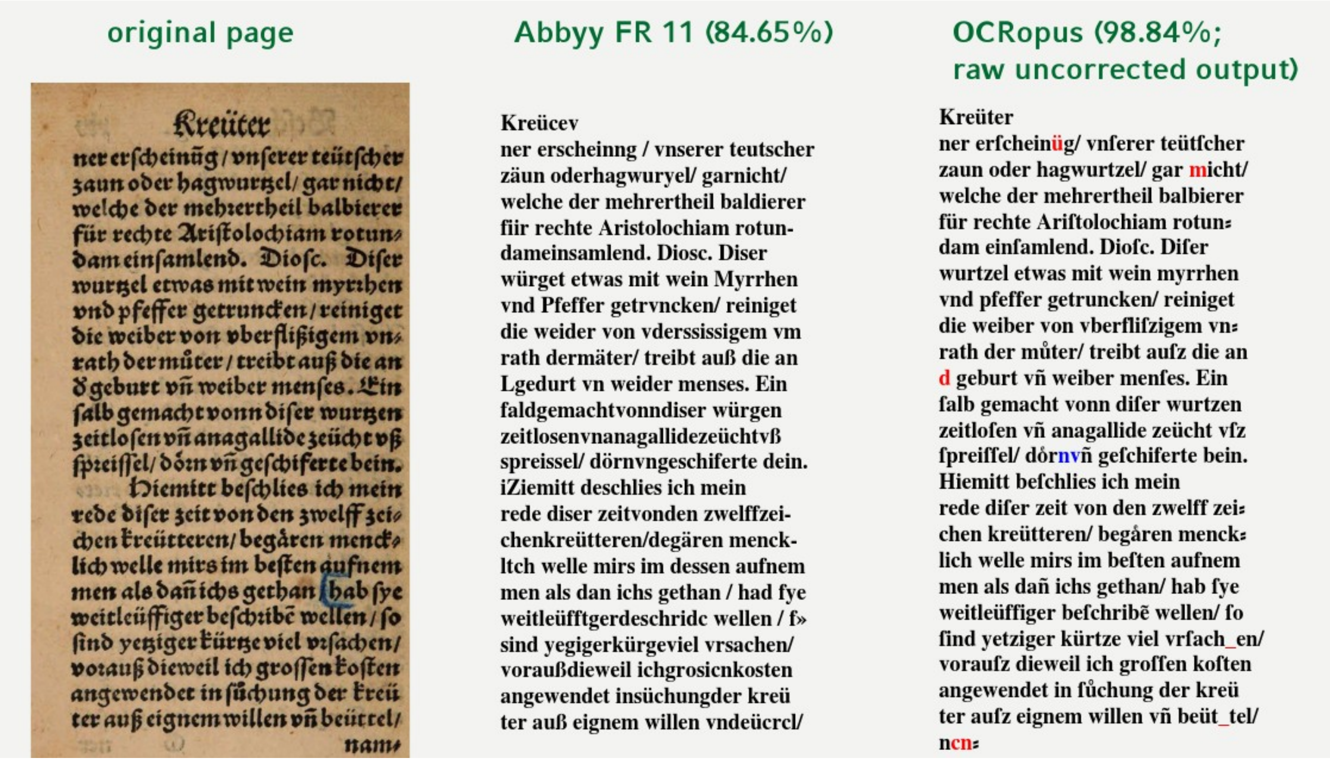
mychars="ʿʹ´ăúáó" becomes
rechars="›‹€†ÿâö" letter for letter.

8 RESULTS

Breuel reported results on machine printed English and Fraktur recognition, back in 2013 (Breuel-LSTM-OCR-ICDAR13.Pdf n.d.).

```
“The LSTM architecture achieved 0:6% character-level test-set error
on English text. When the artificially degraded Fraktur data
set is divided into training and test sets, the system achieves
an error rate of 1:64%. On specific books printed in Fraktur
(not part of the training set), the system achieves error rates of
0:15% (Fontane) and 1:47% (Ersch-Gruber). These recognition
accuracies were found without using any language modelling or
any other post-processing techniques.”
```

Uwe Springemann reported results of 98.84% accuracy (Tutorial.Pdf n.d.). Incidentally, Ocrocis is a easy to use wrapper around Ocropus built for processing complete books. The tutorial gives a detailed description of each of the processing steps.



original page

Kreüter
ner erſcheinūg / vnſerer teütſcher
zaun oder hagwurtzel/ gar nicht/
welche der mehrertheil balbierer
für rechte Ariſtolochiam rotun⸗
dam einſamlend. Dioſc. Diſer
wurtzel etwas mit wein myrrhen
vnd pfeffer getruncken/ reiniget
die weiber von vberflißigem vn⸗
rath der můter/ treibt auß die an
d᷑ geburt vñ weiber menſes. Ein
ſalb gemacht vonn diſer wurtzen
zeitloſen vñ anagallide zeücht vß
ſpreiſſel/ dórn vñ geſchiferte bein.
Hiemitt beſchlies ich mein
rede diſer zeit von den zwelff zei⸗
chen kreütteren/ begåren menck⸗
lich welle mirs im beſten aufnem
men als dañ ichs gethan/ hab ſye
weitleüffiger beſchribē wellen/ ſo
ſind yetziger kürtze viel vrſachen/
vorauß dieweil ich groſſen koſten
angewendet in ſůchung der kreü
ter auß eignem willen vñ beüttel/
nam⸗

Abbyy FR 11 (84.65%)

Kreücev
ner erscheinng / vnserer teutscher
zäun oderhagwuryel/ garnicht/
welche der mehrertheil baldierer
fiir rechte Aristolochiam rotun-
dameinsamlend. Diosc. Diser
würget etwas mit wein Myrrhen
vnd Pfeffer getrvncken/ reiniget
die weider von vderssissigem vm
rath dermäter/ treibt auß die an
Lgedurt vn weider menses. Ein
faldgemachtvonndiser würgen
zeitlosenvnanagallidezeüchtvß
spreissel/ dörnvngeschiferte dein.
iZiemitt deschlies ich mein
rede diser zeitvonden zwelffzei-
chenkreütteren/degären menck-
ltch welle mirs im dessen aufnem
men als dan ichs gethan / had fye
weitleüfftgerdeschridc wellen / f»
sind yegigerkürgeviel vrsachen/
voraußdieweil ichgrosicnkosten
angewendet insüchungder kreü
ter auß eignem willen vndeücrcl/

OCRopus (98.84%;
raw uncorrected output)

Kreüter
ner erfcheinüg/ vnferer teütfcher
zaun oder hagwurtzel/ gar micht/
welche der mehrertheil balbierer
für rechte Ariftolochiam rotun⸗
dam einfamlend. Diofc. Difer
wurtzel etwas mit wein myrrhen
vnd pfeffer getruncken/ reiniget
die weiber von vberflifzigem vn⸗
rath der můter/ treibt aufz die an
d geburt vñ weiber menfes. Ein
falb gemacht vonn difer wurtzen
zeitlofen vñ anagallide zeücht vfz
fpreiffel/ dǒrnvñ gefchiferte bein.
Hiemitt befchlies ich mein
rede difer zeit von den zwelff zei⸗
chen kreütteren/ begåren menck⸗
lich welle mirs im beften aufnem
men als dañ ichs gethan/ hab fye
weitleüffiger befchribẽ wellen/ fo
find yetziger kürtze viel vrfach_en/
voraufz dieweil ich groffen koften
angewendet in fůchung der kreü
ter aufz eignem willen vñ beüt_tel/
ncn⸗

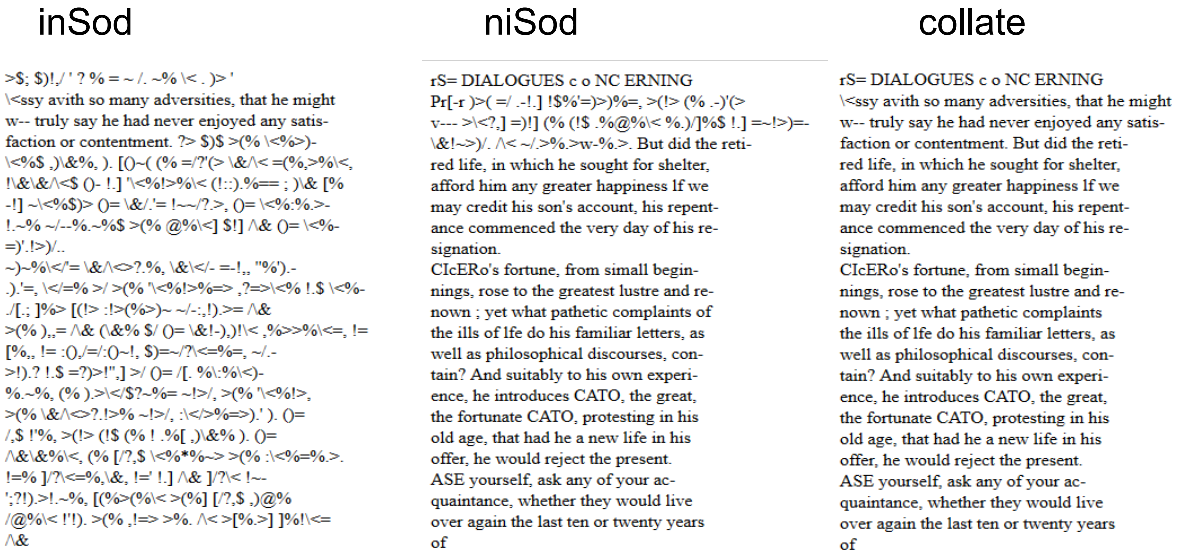
Illustration 5: Results 98.84% accuracy

I do not have numbers for the work I did with the Hume Dialogues (Hume 1779) which was our test vehicle, or the others for that matter, only images. For Hume Dialogues I chose to separate the italics from the normal letters, but chose not to separate the capital letters. Illustration below shows Ocropy results from using the two models and the collated results.

Terminology:

- niSod: Trained model using Normal letters, Italics letters mapped to punctuation Symbols and Over Default model

- inSod: Trained model using Italics letters, Normal letters mapped to punctuation Symbols and Over Default model



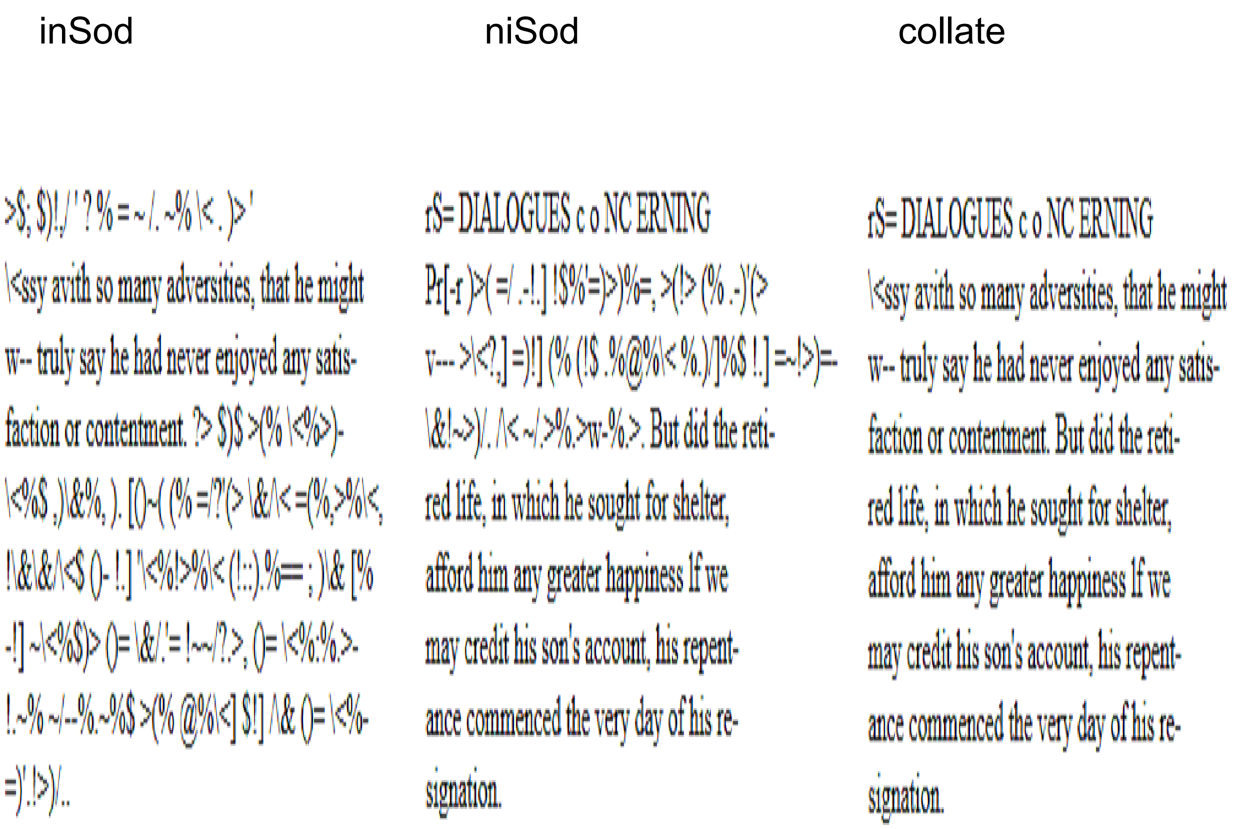
```
>$; $)!,/ ' ? % = ~ /. ~% \< . )> '
\<ssy avith so many adversities, that he might
w-- truly say he had never enjoyed any satis-
faction or contentment. ?> $)$ >(% \<%>)-
\<%$ ,)\&%, ). [()~( (% =/?'(> \&/\< =(%,>%\<,
!\&\&/\<$ ()- !.] "\<%!>%\< (!::).%== ; )\& [%
-!] ~\<%$)> ()= \&/.'= !~~/?.>, ()= \<%:%.>-
!.~% ~/--%.~%$ >(% @%\<] $!] /\& ()= \<%-
=)'.!>)/..
~)~%\</"= \&/\<>?.%, \&\</- =-!,, "%').-
.).'=, \</=% >/ >(% '\<%!>%=> ,?=>\<% !.$ \<%-
./[.; ]%> [(!> :!>(%>)~ ~/-:,!).>= /\&
>(% ),,= /\& (\&% $/ ()= \&!-),)!\< ,%>>%\<=, !=
[%,, != :(),/=/:()~!, $)=~/?\<=%=, ~/.-
>!).? !.$ =?)>!",] >/ ()= /[. %\:%\<)-
%.~%, (% ).>\</$?~%= ~!>/, >(% '\<%!>,
>(% \&/\<>?.!>% ~!>/, :\</>%=>).' ). ()=
/,$ !'%, >(!> (!$ (% ! .%[ ,)\&% ). ()=
/\&\&%\<, (% [/?,$ \<%*%~> >(% :\<%=%.>.
!=% ]/?\<=%,\&, !=' !.] /\& ]/?\< !~-
';?!).>!.~%, [(%>(%\< >(%] [/?,$ ,)@%
/@%\< !'!). >(% ,!=> >%. /\< >[%.>] ]%!\<=
/\&
```

```
rS= DIALOGUES c o NC ERNING
Pr[-r )>( =/ .-!.] !$%'=)>)%=, >(!> (% .-)'(>
v--- >\<?,] =)!] (% (!$ .%@%\< %.)/]%$ !.] =~!>)=-
\&!~>)/. /\< ~/.>%.>w-%.>. But did the reti-
red life, in which he sought for shelter,
afford him any greater happiness lf we
may credit his son's account, his repent-
ance commenced the very day of his re-
signation.
CIcERo's fortune, from simall begin-
nings, rose to the greatest lustre and re-
nown ; yet what pathetic complaints of
the ills of lfe do his familiar letters, as
well as philosophical discourses, con-
tain? And suitably to his own experi-
ence, he introduces CATO, the great,
the fortunate CATO, protesting in his
old age, that had he a new life in his
offer, he would reject the present.
ASE yourself, ask any of your ac-
quaintance, whether they would live
over again the last ten or twenty years
of
```

```
rS= DIALOGUES c o NC ERNING
\<ssy avith so many adversities, that he might
w-- truly say he had never enjoyed any satis-
faction or contentment. But did the reti-
red life, in which he sought for shelter,
afford him any greater happiness lf we
may credit his son's account, his repent-
ance commenced the very day of his re-
signation.
CIcERo's fortune, from simall begin-
nings, rose to the greatest lustre and re-
nown ; yet what pathetic complaints
the ills of lfe do his familiar letters, as
well as philosophical discourses, con-
tain? And suitably to his own experi-
ence, he introduces CATO, the great,
the fortunate CATO, protesting in his
old age, that had he a new life in his
offer, he would reject the present.
ASE yourself, ask any of your ac-
quaintance, whether they would live
over again the last ten or twenty years
of
```

Illustration 6: Collated results for a page from The Hume Dialogues

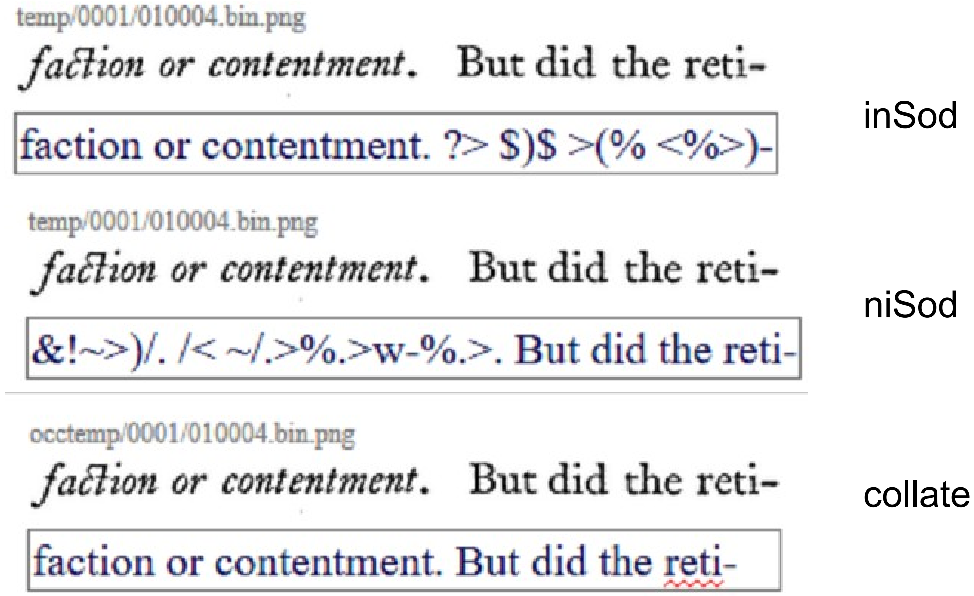
A closer look at the same results is shown below.



inSod	niSod	collate
>$; $)!.,/ ' ? % = ~ /. ~% \< .)>'	rS= DIALOGUES c o NC ERNING	rS= DIALOGUES c o NC ERNING
\<ssy avith so many adversities, that he might	Pr[-r)>(=/ .-!.] !$%'=)>)%=, >(!> (% .-)'(>	\<ssy avith so many adversities, that he might
w-- truly say he had never enjoyed any satis-	v--- >\<?,] =)!] (% (!$.%@%\< %.)/]%$!.] =~!>)=-	w-- truly say he had never enjoyed any satis-
faction or contentment. ?> $)$ >(% \<%>)-	\&!~>)/. /\< ~/.>%.>w-%.>. But did the reti-	faction or contentment. But did the reti-
\<%$,)\&%,). [()~((% =/?'(> \&/\< =(%,>%\<,	red life, in which he sought for shelter,	red life, in which he sought for shelter,
!\&\&/\<$ ()- !.] "\<%!>%\< (!::).%== ;)\& [%	afford him any greater happiness If we	afford him any greater happiness If we
-!] ~\<%$)> ()= \&/.'= !~~/?.>, ()= \<%:%.>-	may credit his son's account, his repent-	may credit his son's account, his repent-
!.~% ~/--%.~%$ >(% @%\<] $!] /\& ()= \<%-	ance commenced the very day of his re-	ance commenced the very day of his re-
=)'.!>)/..	signation.	signation.

Illustration 7: A closer look at part of the page in Hume Dialogues

A close look at a line with font variations is shown below.



```
temp/0001/010004.bin.png
faction or contentment.   But did the reti-
faction or contentment. ?> $)$ >(% <%>)-
```

inSod

```
temp/0001/010004.bin.png
faction or contentment.   But did the reti-
&!~>)/. /< ~/.>%.>w-%.>. But did the reti-
```

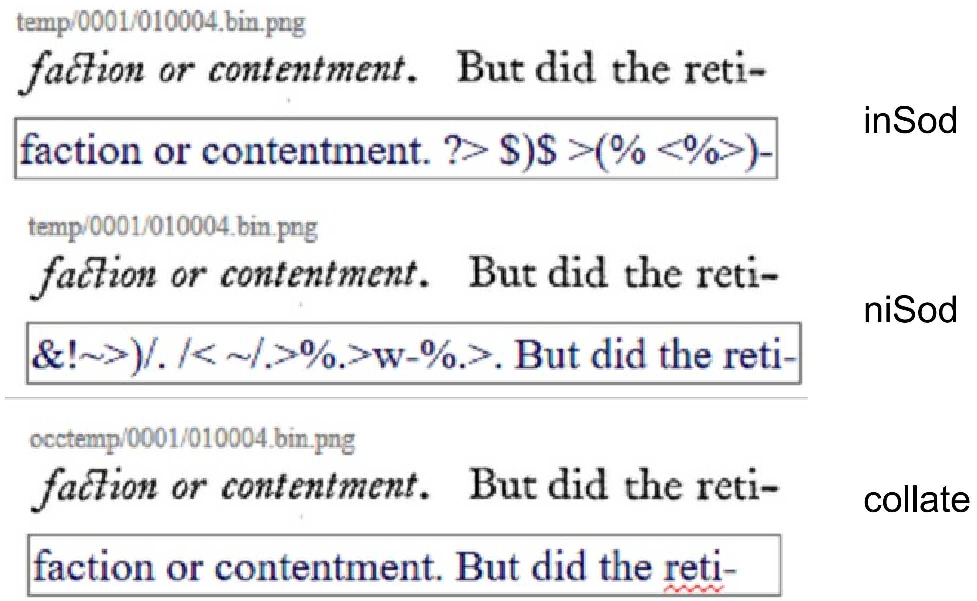
niSod

```
occtemp/0001/010004.bin.png
faction or contentment.   But did the reti-
faction or contentment. But did the reti-
```

collate

Illustration 8: Results for a single line with font variations, Hume Dialogues

A close look at a line without font variations is shown below.



temp/0001/010004.bin.png

ſaction or contentment. But did the reti-

faction or contentment. ?> $)$ >(% <%>)-

temp/0001/010004.bin.png

ſaction or contentment. But did the reti-

&!~>)/. /< ~/.>%.>w-%.>. But did the reti-

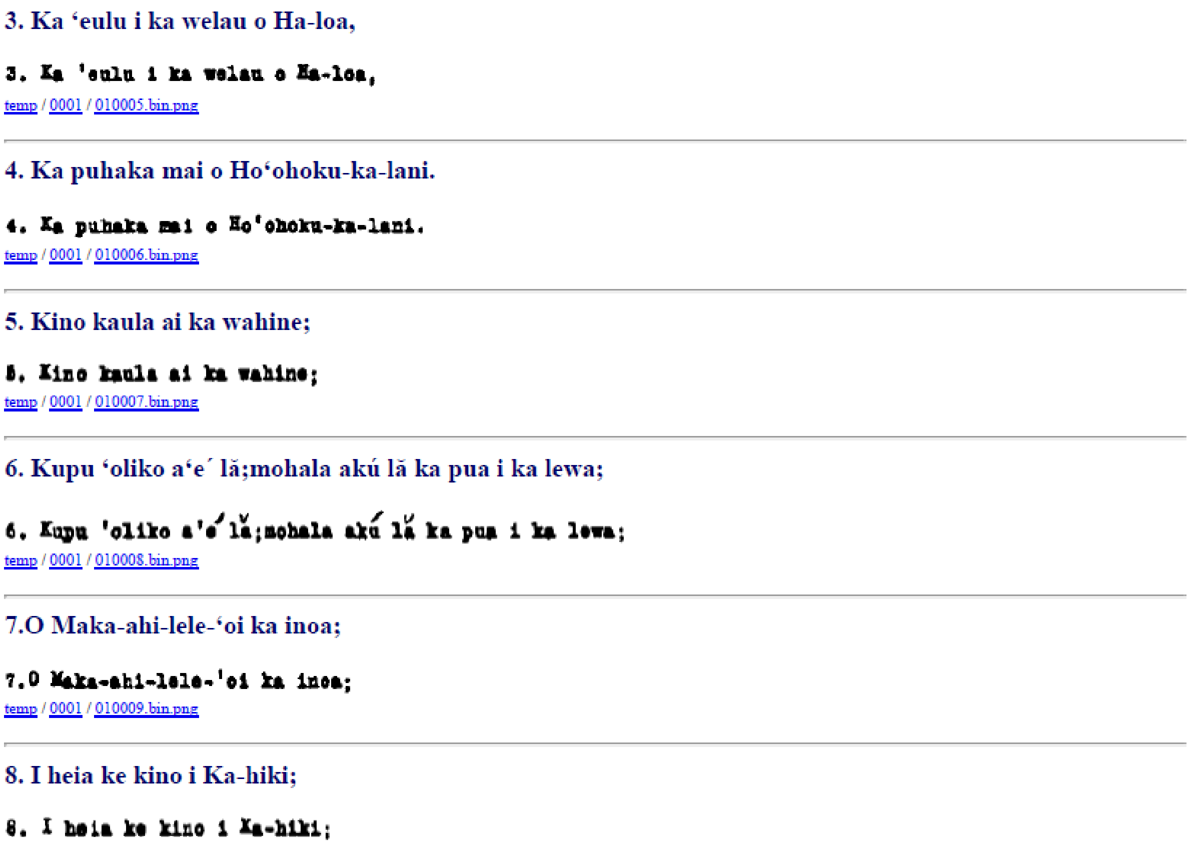
occtemp/0001/010004.bin.png

ſaction or contentment. But did the reti-

faction or contentment. But did the reti-

Illustration 9: Results for a line without any font variations, Hume Dialogues

A page with special characters (Hawaiian).



3. Ka ‘eulu i ka welau o Ha-loa,

3. Ka 'eulu i ka welau o Ha-loa,

temp / 0001 / 010005.bin.png

4. Ka puhaka mai o Ho‘ohoku-ka-lani.

4. Ka puhaka mai o Ho'ohoku-ka-lani.

temp / 0001 / 010006.bin.png

5. Kino kaula ai ka wahine;

5. Kino kaula ai ka wahine;

temp / 0001 / 010007.bin.png

6. Kupu ‘oliko a‘e´ lă;mohala akú lă ka pua i ka lewa;

6. Kupu 'oliko a'é lǎ;mohala akú lǎ ka pua i ka lewa;

temp / 0001 / 010008.bin.png

7.O Maka-ahi-lele-‘oi ka inoa;

7.O Maka-ahi-lele-'oi ka inoa;

temp / 0001 / 010009.bin.png

8. I heia ke kino i Ka-hiki;

8. I heia ke kino i Ka-hiki;

Illustration 10: Results for a page with special characters, Hawaiian.

9 Self Assessment

Answer the following questions, answers are in Appendix E.

1. Which statement below best describes Ocropy?
   a) Recurrent Neural Network (RNN) based text prediction software
   b) Long Short Term Memory architecture based text segmentation software
   c) Long Short Term Memory architecture based collection of programs for text recognition
   d) Recurrent Neural Network based image binarization software

2. What is the default structure of the LSTM in Ocropy
   a) input size 16, internal state units 64
   b) input size 32, internal state units 96
   c) input size 48, internal state units 100
   d) input size 16, internal state units 100

3. What should be the minimum resolution of the input image?
   a) 300 dpi
   b) 96 dpi
   c) 192 dpi
   d) 256 dpi

4. Ocropy is
   a) Is based on shape models. It detects regions of text lines.
   b) Font based. Every font has to be modeled.
   c) Is based on patterns in time series. So it is context aware.
   d) Both a) and c) above.

5. How does Ocropy take into account varying writer speed and non-uniform skew?
   a) Uses skew steps to calculate skew
   b) Scales the lines using skew steps
   c) Flattens the line onto a straight line
   d) uses a normalization process to scale the line

6. The default model file is
   a) latin-default.pyrnn.gz
   b) en-default.pyrnn.gz
   c) in-default.pyrnn.gz
   d) us-default.pyrnn.gz

7. The character set of the default model
   a) has 157 characters and can be changed
   b) has 157 characters and can add only special characters
   c) has 157 characters and cannot be changed
   d) has 157 alphanumeric characters

8. Ground truth text files, *.gt.txt can be produced by
   a) editing temp-correction.html produced by ocropus-gtedit and extract using the same
   b) copy the *.txt files produced by the predictor ocropus-rpred to *.gt.txt and edit manually

c) A and B
d) neither A nor B

9. With the lines produced by Ocropy using the default model, for best results compose the training set with
   a) all lines from all pages
   b) only the lines that shows errors in Ocropy predictions
   c) lines that Ocropy predicts correctly
   d) all lines from one page only

10. For best results, choose pages or images that
    a) contain ALLCAPS letters only
    b) contain normal letters only
    c) contain both ALLCAPS and normal letters
    d) contains numerical letters only

11. For best results, choose
    a) training vectors that are from one page only
    b) training vectors that evens out frequency of occurrence of letters
    c) training vectors from the first page of every chapters
    d) training vectors from the last page of every chapters

12. Binarization chooses if a pixel value is
    a) 0 or maximum
    b) noise level
    c) information
    d) noise or information

13. Which statement best describes effect of skew on segmentation results?
    a) More accurate skew calculation gives best segmentation results
    b) A skew of 0 gives best segmentation results
    c) Less accurate the skew calculation, less segments found
    d) More accurate the skew calculations, more segments found

14. For ocropus-rpred, the character set is defined in
    a) /usr/local/lib
    b) chars.py
    c) ocrolib.py
    d) none of the above

15. Special characters are
    a) characters that are not part of the English alphabet
    b) characters that are not part of the English alphabet or numbers
    c) characters that are not alphanumeric or symbols
    d) characters that are not part of the 157 characters used by the default model

16. When you have special characters
    a) Use the -c option during training and declare the special characters in chars.py
    b) Use the default model and the -c option during training

c) Use the -c option during training and set PYTHONIOENCODING=”UTF-8”
d) Set PYTHONIOENCODING=”UTF-8” and declare the special characters in chars.py

10 Bibliography

Breuel, Thomas
2008 The OCRopus Open Source OCR System, vol.6815.

Breuel, Thomas M.
2008 The OCRopus Open Source OCR System. *In* . Berrin A. Yanikoglu and Kathrin Berkner, eds. P. 68150F–68150F–15. http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=812144, accessed November 30, 2017.

Breuel-LSTM-OCR-ICDAR13.Pdf
N.d. http://staffhome.ecm.uwa.edu.au/~00082689/papers/Breuel-LSTM-OCR-ICDAR13.pdf, accessed November 12, 2017.

Hume, David
1779 Dialogues Concerning Natural Religion. London: : [s.n.]. http://archive.org/details/dialoguesconcern1779hume, accessed December 15, 2017.

Implementation of Resizing Codec by ChWick · Pull Request #277 · Tmbdev/Ocropy
N.d. https://github.com/tmbdev/ocropy/pull/277, accessed December 14, 2017.

OCRopus
2017 Wikipedia. https://en.wikipedia.org/w/index.php?title=OCRopus&oldid=787599524, accessed November 11, 2017.

Practical Expercience with OCRopus Model Training : Arbeitsgruppe Kommunikationssysteme
N.d. https://comsys.informatik.uni-kiel.de/lang/de/res/practical-expercience-with-ocropus-model-training/, accessed November 30, 2017.

Tmbdev/Ocropy
N.d. GitHub. https://github.com/tmbdev/ocropy, accessed December 11, 2017.

Tom
2017a[2015] Clstm: A Small C++ Implementation of LSTM Networks, Focused on OCR. Jupyter Notebook. https://github.com/tmbdev/clstm, accessed November 12, 2017.
2017b[2014] Ocropy: Python-Based Tools for Document Analysis and OCR. Jupyter Notebook. https://github.com/tmbdev/ocropy, accessed November 12, 2017.

Training an Ocropus OCR Model
N.d. http://www.danvk.org/2015/01/11/training-an-ocropus-ocr-model.html, accessed November 30, 2017, a.
N.d. http://www.danvk.org/2015/01/11/training-an-ocropus-ocr-model.html, accessed November 30, 2017, b.

Tutorial.Pdf
N.d. http://cistern.cis.lmu.de/ocrocis/tutorial.pdf, accessed December 15, 2017.

Winder, Amy, Tim Andersen, and Elisa Barney Smith
2011 Extending Page Segmentation Algorithms for Mixed-Layout Document Processing. Computer Science Faculty Publications and Presentations. http://scholarworks.boisestate.edu/cs_facpubs/19, accessed December 11, 2017.

Workflow.Vue
N.d. https://rawgit.com/tmbdev/ocropy/master/doc/workflow.html, accessed November 15, 2017.

11 Appendix A: Pre-processing and Recognition

```
#!/bin/bash -e

BASE=$(dirname $0)
ESCALE=0.1
ZOOM=0.80
SKEWSTEPS=4
NSEPS=300
NLINES=900
SCALE=39.47
#SEPWIDEN=20

echo " BASE = $BASE"
echo " escale = $ESCALE"
echo " zoom = $ZOOM"
echo " skewsteps = $SKEWSTEPS"
echo " Max black column seperators = $NSEPS"
echo " Max lines = $NLINES"
image=$BASE/png/103_002_108_04_19061008.png


rm -rf temp
# Binarize image
echo ocropus-nlbin $image -z $ZOOM -o temp -n
ocropus-nlbin $image -z $ZOOM -o temp -n
# Segment image
echo ./ocropus-gpageseg -n -b --maxseps $NSEPS --maxlines $NLINES --scale
$SCALE 'temp/????.bin.png' --debug
./ocropus-gpageseg -n -b --maxseps $NSEPS --maxlines $NLINES --scale $SCALE
'temp/????.bin.png' --debug
# Predict lines
# Default
echo ocropus-rpred -n 'temp/????/??????.bin.png'
ocropus-rpred -n 'temp/????/??????.bin.png'
# Collate lines
echo ocropus-hocr 'temp/????.bin.png' -o temp.html
ocropus-hocr 'temp/????.bin.png' -o temp.html
# Visualize results
echo ocropus-visualize-results temp
ocropus-visualize-results temp
# Create ground truth files
echo ocropus-gtedit html temp/????/??????.bin.png -o temp-correction.html
ocropus-gtedit html temp/????/??????.bin.png -o temp-correction.html

echo "to see recognition results, type: firefox temp.html"
echo "to see correction page, type: firefox temp-correction.html"
echo "to see details on the recognition process, type: firefox
temp/index.html"
```

12 Appendix B: Model Evaluation

```
#!/bin/bash -e

BASE=$(dirname $0)
MODELS=$BASE"/models/img2"
TESTS=$BASE"/testdata/testimg2"
PNGFILES=$TESTS"/??????.bin.png"
GTFILES=$TESTS"/??????.gt.txt"

echo
echo "Model files are in $MODELS"
echo "Test files are in $TESTS"
echo "PNG files are $PNGFILES"
echo "GT  files are $GTFILES"
echo

if [ -f err.log ]
then
    \rm err.log
fi
touch err.log

if [ -f err.txt ]
then
    \rm err.txt
fi
touch err.txt

gzfiles=$MODELS"/*.gz"
for modelfile in $gzfiles
do
    modelno=`echo $modelfile | sed -e "s/[a-zA-Z/.-]//g"`
    echo >> err.log
    echo "Running model file $modelfile"
    echo "results using model file $modelfile" >> err.log
    echo ocropus-rpred -n $PNGFILES -m $modelfile >> err.log
    ocropus-rpred -n $PNGFILES -m $modelfile >> err.log
    echo ocropus-errs -e $GTFILES >> err.log 2>&1
    errstr=`ocropus-errs -e $GTFILES`
    echo $errstr >> err.log
    errn=`echo "${errstr}" | grep errors | sed -e "s/[a-z%]//g"`
    errp=`echo "${errstr}" | grep errnomiss | sed -e "s/[a-z%]//g"`
    echo $modelno $errn $errp
    echo "$modelno $errn $errp" >> err.txt
    echo ocropus-econf $GTFILES
    ocropus-econf $GTFILES
done
```

13 Appendix C: Diagnostic messages

```
ocropus-nlbin ./../work_hawaii/png/103_002_108_04_19061008.png -o temp -n
INFO:  # ./../work_hawaii/png/103_002_108_04_19061008.png
INFO:  === ./../work_hawaii/png/103_002_108_04_19061008.png 1
INFO:  flattening
INFO:  estimating skew angle
INFO:  estimating thresholds

INFO:  rescaling
INFO:  ./../work_hawaii/png/103_002_108_04_19061008.png lo-hi (0.28 0.60) angle  0.0
INFO:  writing
./ocropus-gpageseg -n -b --maxseps 300 --maxlines 900 temp/????.bin.png
INFO:
INFO:  ########## ./ocropus-gpageseg -n -b --maxseps 300 --maxlines 900 temp/?
INFO:
INFO:  temp/0001.bin.png
INFO:  scale 45.475268
INFO:  computing segmentation
INFO:  computing column separators

INFO:  computing lines

INFO:  propagating labels
INFO:  spreading labels
INFO:  number of lines 160
INFO:  finding reading order
INFO:  writing lines

INFO:     139  temp/0001.bin.png 45.5 140
```

14 Appendix D: Example Procedure

- Convert the image to .png format. The ImageMagic software comes equipped with an easy to use utility called convert. Converting from one format to another is as easy as using the corresponding file extension. For example, to convert from .tif to .png run,

  ```
  > convert test.tif test.png
  ```

- Create the training set. Ocropy needs to be trained on the new “font” before it can successfully recognize it.

  Ocropy uses the Long Short Term Memory (LSTM) type of recurrent neural network as the recognition engine. Ocropy installation by default is equipped with a LSTM that is trained on all the alphanumeric characters for 100000 iterations. This model file is called en-default.pyrnn.gz. Using this, Ocropy will be able to recognize all alphanumeric characters in our image, although the recognition will be incorrect.

  a. Edit the run script and change the name of the image file

  ```
  > image=$BASE/png/Mele/MS2x-0.png
  ```

  b. Make sure the Ocropus-rpred is invoking the default model.

  ```
  > ocropus-rpred -n 'temp/????/??????.bin.png'
  ```

  OR

  ```
  > ocropus-rpred -n 'temp/????/??????.bin.png' -m en-default.pyrnn.gz
  ```

  c. Run run-test

  ```
  > ./run-test
  ```

  d. Ocropus extracts the lines of text in the image and stores them in a directory called 0001 in the temp directory. This directory will contain an image file for each text line and the ascii text that it recognizes the line of text to be. This text will be wrong. Open the image files one by one and correct the corresponding ascii text in the text file. Rename each of the text file from .txt to .gt.txt extension. You have created a training set.

- Separate the font variations. To have Ocropy recognize the new font correctly we have to train it on all the variations of the font individually, such as the italics, the caps and the normal letters.

  a. Separate the italics, the caps and the normal letters from the training set and store them in separate directories. You may have to clip the image and the corresponding ascii text to isolate the font variations. Name the directories train_italics, train_caps and train_normal respectively.

- Repeat steps 2 and 3 for several such images. Choose images that contain all the font variations. For every two such lines extracted, store the third line in a different directory for testing the performance of the model. Call these directories test_italics, test_caps and test_normal, respectively.

- High performance computing for training.Use the high performance computers to run the training of the LSTM. Copy the training directories from step 3 over to the high performance computing clusters.

  ```
  > scp -r train_italics $username@uhhpc1.its.hawaii.edu :lus
  > scp -r train_caps $username@uhhpc1.its.hawaii.edu :lus
  > scp -r train_normal $username@uhhpc1.its.hawaii.edu :lus
  ```

- Train on top of the default model.

```
> cd train_italics
> ocropus-rtrain --load en-default.pyrnn.gz -o hume-model-i *.png > result.log 2>&1
> cd train_caps
> ocropus-rtrain --load en-default.pyrnn.gz -o hume-model-c *.png > result.log 2>&1
> cd train_normal
> ocropus-rtrain --load en-default.pyrnn.gz -o hume-model-n *.png > result.log 2>&1
```

This is the first round of training, so train for 5000 - 10000 iterations, no more. Ocropus adds the iteration number to the name of the file and saves the model every 1000 iterations. For example, the italics model after 5000 iterations will be called hume-model-i-00105000.pyrnn.gz.

- Repeat step 6 for the next font variation. For example for the italics model produced in step 6, you train with the caps data.

  ```
  > cd train_italics
  > ocropus-rtrain --load hume-model-n-00105000.pyrnn.gz -o hume-model-ion *.png >
  result.log 2>&1
  > cd train_caps
  > ocropus-rtrain --load hume-model-i-00105000.pyrnn.gz -o hume-model-coi *.png >
  result.log 2>&1
  > cd train_normal
  > ocropus-rtrain --load hume-model-c-00105000.pyrnn.gz -o hume-model-noc *.png >
  result.log
  ```

  This is the second round of training, once again train for 5000-10000 iterations, no more. Ocropus once again adds the iteration number to the name of the file and saves the model every 1000 iterations. For example, the italics model after 5000 iterations will be called, in this case, hume-model-ion-00110000.pyrnn.gz. “ion” stands for italics over normal, “coi” stands for caps over italics and “noc” stands for normal over caps.

- Repeat step 6 for the third and final font variation. For example for the italics model produced in step 7, you once again train with the caps data.

  ```
  > cd train_italics
  > ocropus-rtrain --load hume-model-noc-00110000.pyrnn.gz -o hume-model-ionoc *.png
  > result.log 2>&1
  > cd train_caps
  > ocropus-rtrain --load hume-model-ion-00110000.pyrnn.gz -o hume-model-coion *.png
  > result.log 2>&1
  > cd train_normal
  > ocropus-rtrain --load hume-model-coi-00110000.pyrnn.gz -o hume-model-nocoi *.png
  > result.log
  ```

  This is the final round of training, this time let it train for 20000-40000 iterations. Ocropus once again adds the iteration number to the name of the file and saves the model every 1000 iterations. For example, the italics model after 15000 iterations will be called, in this case, hume-model-ionoc-00125000.pyrnn.gz.

- Copy the three models over from the high performance clusters, the ones every 1000 iteration that Ocropus has saved to disk. Run the performance test for each of the models. Use the test directories produced in step 4.

  ```
  > cd test_italics
  > ocropus-rpred -n $PNGFILES -m $modelfile >> err1.log
  > ocropus-errs -e $GTFILES >> err1.log 2>&1
  ```

  where PNGFILES and GTFILES refers to the directory containing images of the lines. modelfile contains the path to the name of the model file being tested.

- Save the model file that returns the least number of errors during the performance testing. There may be several models that return the least number of errors.
- Change run-test script to invoke one of the models during the recognition step, ocropus-rpred.
  ```
  ocropus-rpred -n 'temp/????/??????.bin.png' -m $BASE/models/hume-model-ionoc-00135000.pyrnn.gz
  ```
  Store the resulting text file, call it ionoc.txt. Repeat for the other two models and store the results as coion.txt and nocoi.txt.
  ```
  ocropus-rpred -n 'temp/????/??????.bin.png' -m $BASE/models/hume-model-coion-00129000.pyrnn.gz
  ocropus-rpred -n 'temp/????/??????.bin.png' -m $BASE/models/hume-model-nocoi-00144000.pyrnn.gz
  ```

- Write a script that parses each of these text files and chooses the “word that makes the most sense” at every word location.

15 Appendix E: Self Assessment answers

1. C
   Although all the answers are true, Ocropy is a collections of programs that implements a Long Short Term Memory for text recognition.

2. C
   Ocropy default input size is 48, internal state units is 100

3. A
   Ocropy expects a minimum input image resolution of 300 dpi.

4. D
   Ocropy uses shape models and detects regions of text lines. Thus it is based on patterns in time series and is context aware.

5. D
   Ocropy uses a normalization process to scale the line

6. B
   Ocropy default model file is called en-default.pyrnn.gz

7. D
   The default Ocropy model has 157 characters and cannot be changed.

8. C
   You can edit temp-correction.html or edit the .txt file manually to produce the .gt.txt file.

9. B
   For best results, use only the lines that shows errors in Ocropy predictions

10. C
    Contain both ALLCAPS and normal letters, assuming one model is being employed for both.

11. B
    Repeat training vectors so as to even out frequency of occurrence of letters

12. A
    Binarization calculates if a pixel value is 0 or maximum value.

13. A
    Best segmentation results are seen when columns are parallel and to the edges. The skew calculations have to be accurate and the resulting skew in the page closer to 0.

14. B
    chars.py

15. D
    Special characters are characters outside of all the 157 characters used by the default model, which is ascii and xsymbols and French and German.

16. A
    When you have special characters, you use the ocropus-rtrain -c or -codec option during training to generate a new codec. You also declare the special characters in the chars.py for use by ocropus-rpred.