# Code Documentation

**OCR.**inspect\_pytesseract\_ROI\_recognition(cell, demo\_picture, config, col = **None**):

Takes a raw image and return the ROI cognized by pytesseract in red rectangles.

cell

* Input picture

config

* the configuration for pytesseract engine.

demo\_picture

* The back ground picture for demonstration.

col

* the column number, determines whether the engine will be configured to recognize integer or etc.

General

* Use opencv, pytesseract (LSTM, CNN) to segment and OCR pages.
  + Use transformations/ noise models to generate synthetic noise
  + Optical models are useful to synthesize erosion type of noise.
* Use NLTK + rf, chunk processing, GCP VM, census data to train gender classifier
* Use CNN or models to correct spelling, restrict answers to a name/ place dictionary.

0.0.11

Fixed bugs:

The get\_mask() method does not perform well on new samples because of sizing issue.

Now resizing all inputs to (3663, 2831)

0.0.11.1

Another fix: imutils.rezie() -- need to specify the width and the height, otherwise reversed and the result will be wrong.

0.0.12

Committed on 2020/11/17

1) Post class for post processing

- cleaned census name strings. They will be used as a dictionary for OCR post processing processes.

- chunk process

- create meta data generation fuction.

2) Pytesseract behavior inspection utilities

- draw the ROI boxes created by pytesseract.

0.0.13

2020/11/17

1. fixed a character segmentation problem (figure 1-8.) in pre-processing.
   1. Bug: vertical dilation is too strong that we cannot delete
   2. Solution: reduce dilation kernel in the vertical direction.
   3. Note: tesseract is good at segmenting lines with -psm 6, so it might not be that big an issue.

Figure1. dilation input for contour detection



Figure2. contour detection box (green): elements from 2 lines detected as because of detection in figure .



Figure 3. output from noise deletion: failure to detect the noise from the previous line because of



Figure 4. Pytesseract takes input from figure 3 recognize words from the last line.



Figure 5 – 8 Results after reducing vertical dilation by half:







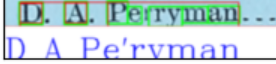


1. Adding deletion of non-character/ non-contour into pre-processing, in addition to masks. Some characters are erroneously deleted if they have disconnected pieces (figure 10)
   1. Figure 9 dilation results



Figure 10 tesseract input





* 1. Figure 12 corrections

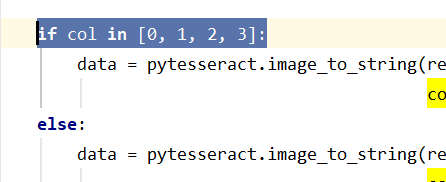








0.0.14

1. Pre-processing: Adding deskew, 2d only.
   * **def** two\_dimensional\_deskew(raw\_colored, display = **False**):
2. New/ Old problems with page segmentation:
   * Page segmentation still missing horizontal lines.
     + E.g. **"OR\_test\_departs\_3.jpg"**
   * Code not compatible with changes in the number of columns
3. Pytesseract problem:
   * 
   * Not reading the space
4. Pytesseract problem
   * OCR.page\_to\_DF()
   * 
   * This needs to be corrected for page with less columns.
5. To do:
   * Synthetic noise (ink, thick ink pattern (use a pen to write times new roman)
   * Try disabling the dictionary check feature in Tesseract input parameters.
     + This fixes autocorrection error that creates wrng output
   * Compare with direct convolution.

2020/11/14

Summary

OCR augmentation methods

* Post-process:
  + train LSTM post processing algorithm with a restrictive dictionary (e.g. first names in the 19th century)
* OCR
  + train/ augment LSTM OCR engine with a perturbated train set generated through AR algorithms
  + Pre-processing
    - Thresholding, erode, dilate
    - Segmentation methods:
      * Character: contour detection, restrictive bounding masks.
      * Page segmentation: customized tensors

Problems

1. Digits not recognized correctly (missing digits)
   1. Reason 1: some digits look like alphabet. E.g. 1 looks like “l”
      1. Maybe use customized train file.
   2. Reason 2: maybe because of some segmentation problem?
      1. Explore character segmentation or use other variations of pytesseract.
      2. 2020/11/14 Overlay segmentation mask in OCR.ROI\_to\_text() with original input, drawing, making clear the rectangle border.
2. Overall segmentation problem—character are not separate from each other, but most importantly for the digits.
3. Need to rewrite codes: delete rectangle if it intersects with the top.

2020/11/14-2

1. Establish a string cleaning meathod
   1. Fuzzy matching with reference to a dictionary
   2. Restrc
2. New segmentation problem: page title/ state name not on top of the page.
   1. Need to recognize title by the size of the block.
   2. Maybe check if tesseract’s regional proposal works.
   3. Or, first of all, use the old method (detect large blocks on the page with titles in two sections first.)

# Process Documentation

Segmentation, Denoise, OCR result

Figure 0. Page segmentation result

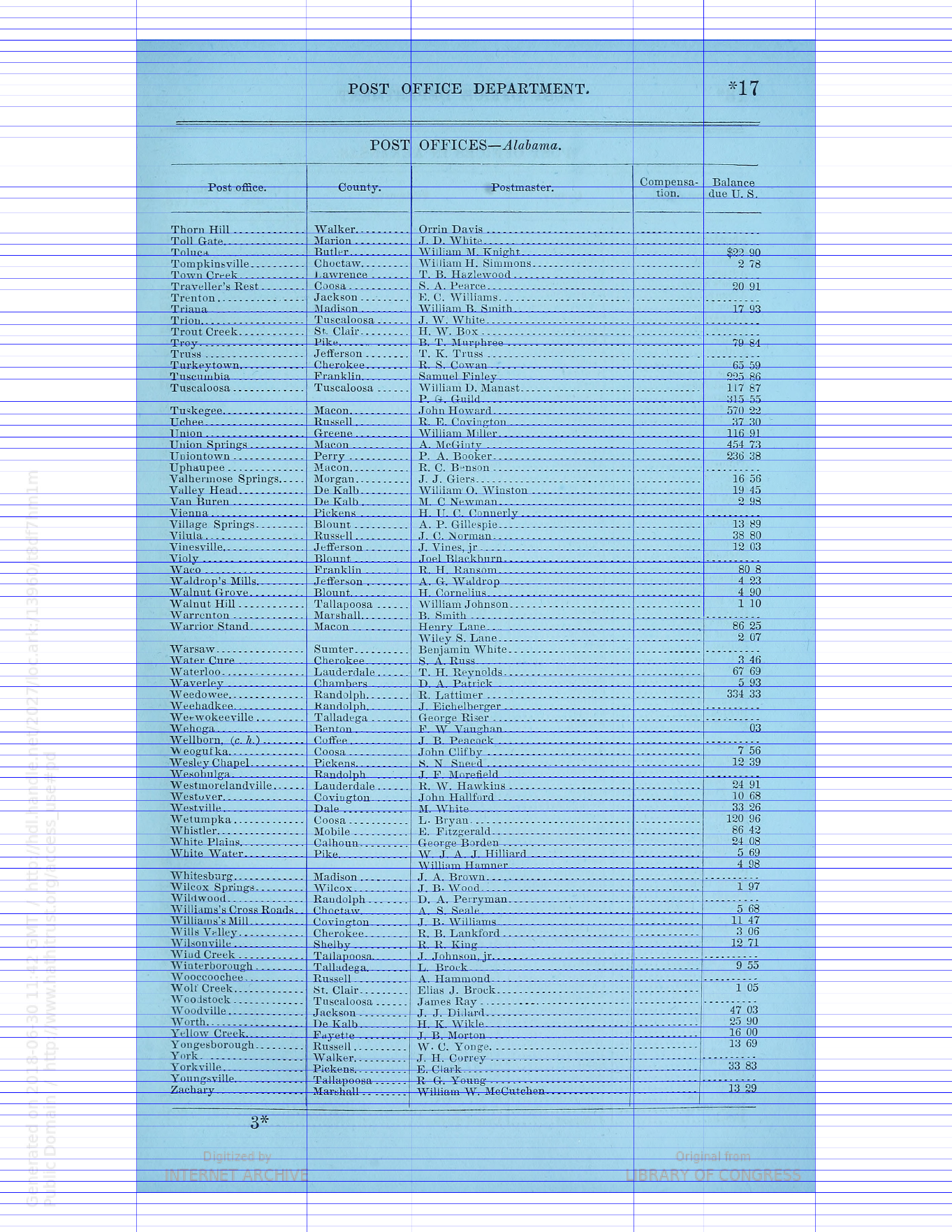


Figure 1. denoised input

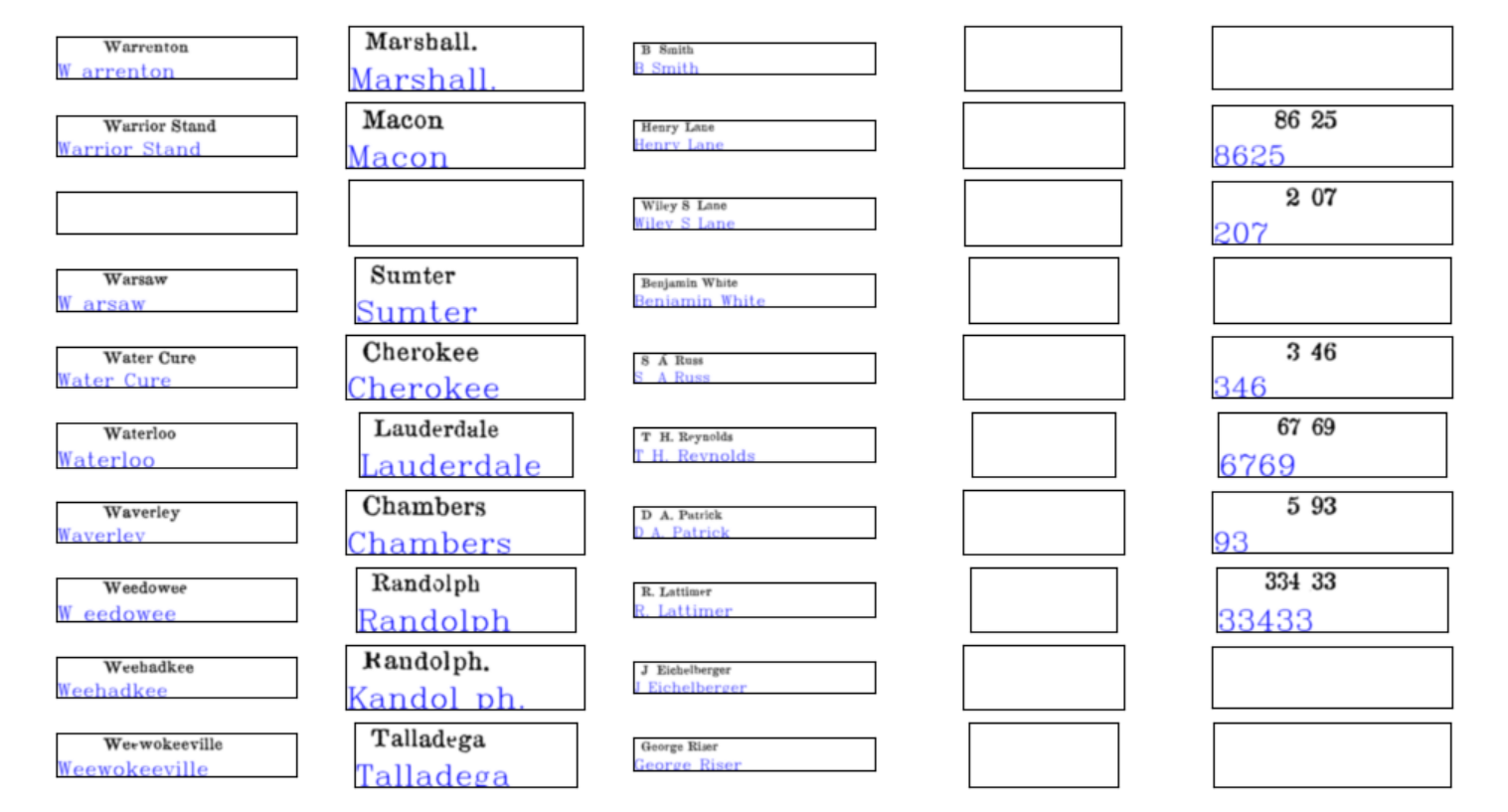
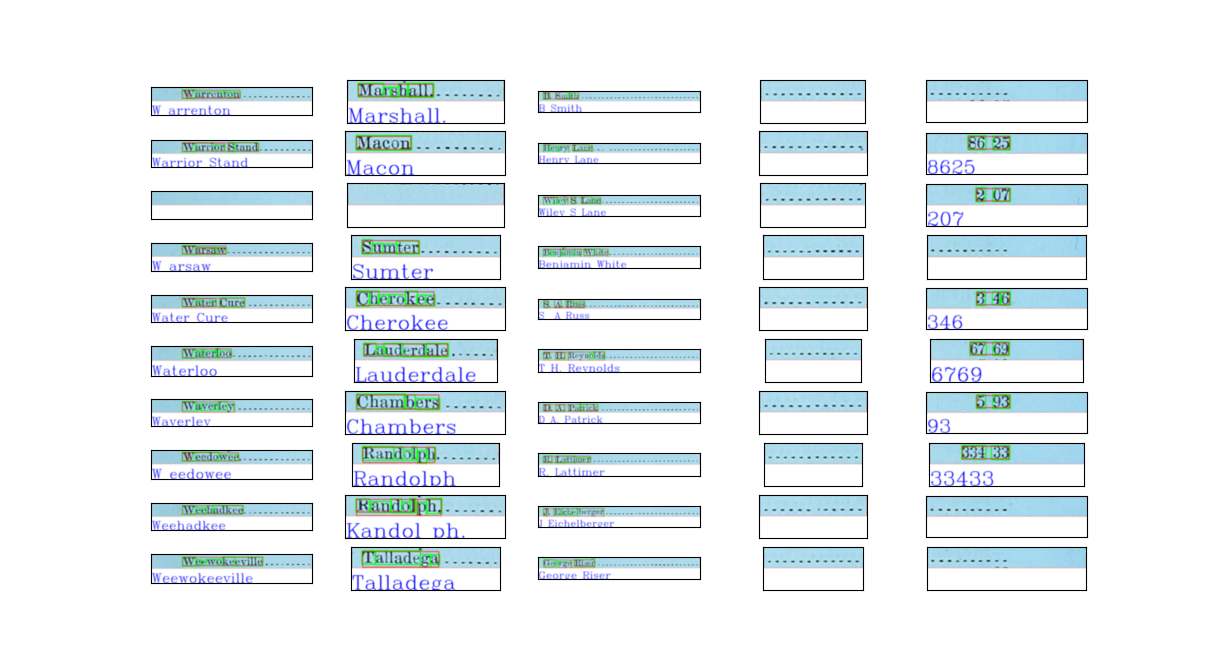


Figure 3 OpenCV contour segmentation mask (green) + pytesseract masked segmentation (red)



# Research Notes

Section 2. Pytesseract Properties

1. Recognition Failure

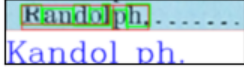
Def. Pytesseract behave poorly on certain “obvious” patterns that are partially eroded. We have enough information in an eroded character pattern, yet it fails to recognize.

Excerpts from Figure 1 above.





These are perhaps not segmentation issues:





Solutions

**-c tessedit\_debug\_fonts=1**

Setting the font debugging parameter does not do anything.

1. Preserve Space

**def** inspect\_pytesseract\_ROI\_recognition(img\_list, demo\_picture)

Section 1. OCR performance

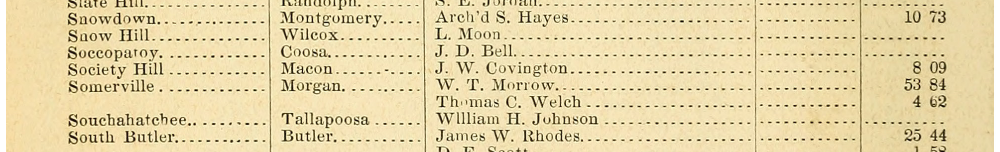


Figure 1-2 Preserve space VS not preserving space for col 0-3

Figure 1.

**if** col **in** [0, 1, 2, 3]:  
 data = pytesseract.image\_to\_string(result, lang=**'eng'**, config=**'--psm 6 -c tessedit\_char\_blacklist=0123456789 -c preserve\_interword\_spaces=1'**)  
**else**:  
 data = pytesseract.image\_to\_string(result, lang=**'eng'**, config=**'--psm 6 outputbase digits -c preserve\_interword\_spaces=1 -c tessedit\_char\_blacklist=.'**)

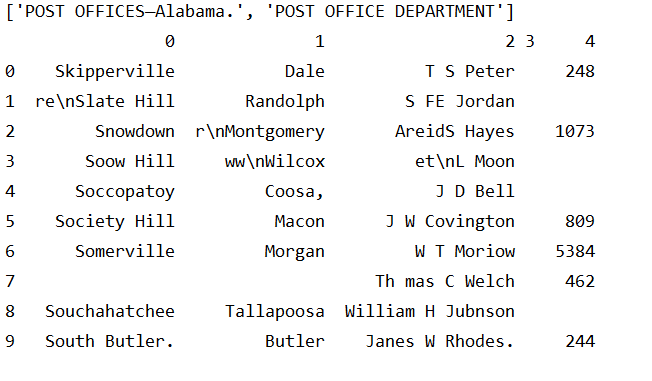
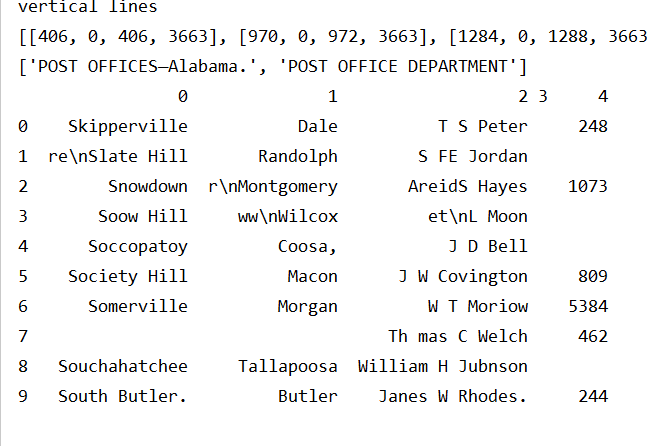
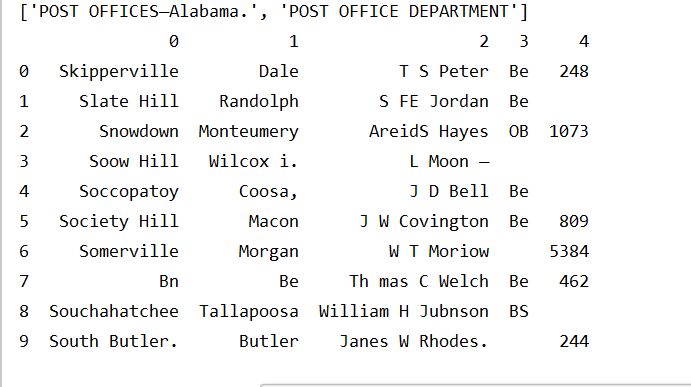


Figure 2. Not preserving space.

**if** col **in** [0, 1, 2, 3]:  
 data = pytesseract.image\_to\_string(result, lang=**'eng'**, config=**'--psm 6 -c tessedit\_char\_blacklist=0123456789'**)  
**else**:  
 data = pytesseract.image\_to\_string(result, lang=**'eng'**, config=**'--psm 6 outputbase digits -c preserve\_interword\_spaces=1 -c tessedit\_char\_blacklist=.'**)



**if** col **in** [0, 1, 2, 3]:  
 data = pytesseract.image\_to\_string(result, lang=**'eng'**, config=**'--psm 7 -c tessedit\_char\_blacklist=0123456789'**)  
**else**:  
 data = pytesseract.image\_to\_string(result, lang=**'eng'**, config=**'--psm 7 outputbase digits -c preserve\_interword\_spaces=1 -c tessedit\_char\_blacklist=.'**)



<http://www.cse.psu.edu/~deh25/post/Timeline_files/US-Official_Register.html>

Chunk Processing and Map Reduce:

<https://pythonspeed.com/articles/chunking-pandas/>

Gender Prediction:

<https://www.geeksforgeeks.org/python-gender-identification-by-name-using-nltk/>

<http://www.digitalhumanities.org/dhq/vol/9/3/000223/000223.html>

# Pre-processing

Correct 3d distortion

<https://www.learnopencv.com/image-alignment-feature-based-using-opencv-c-python/>

# Pytesseract

Page segmentation modes:

0 Orientation and script detection (OSD) only.

1 Automatic page segmentation with OSD.

2 Automatic page segmentation, but no OSD, or OCR.

3 Fully automatic page segmentation, but no OSD. (Default)

4 Assume a single column of text of variable sizes.

5 Assume a single uniform block of vertically aligned text.

6 Assume a single uniform block of text.

7 Treat the image as a single text line.

8 Treat the image as a single word.

9 Treat the image as a single word in a circle.

10 Treat the image as a single character.

11 Sparse text. Find as much text as possible in no particular order.

12 Sparse text with OSD.

13 Raw line. Treat the image as a single text line,

bypassing hacks that are Tesseract-specific.

More parameters at the end of this document (Java)

<https://github.com/cbrtekstraktor/cbrTekStraktor/blob/f1568c218b78d114460b5a52d26e182669fac8ec/src/ocr/cmcOCRController.java#L1907>

How to retrain tesseract

<https://tesseract-ocr.github.io/tessdoc/Training-Tesseract.html>

# Spelling Correction and Census Matching

* Denoise Autoencoders
  + https://github.com/Ezajac/denoising-autoencoder
    - Pretrained model
  + Explanation 1
    - <https://towardsdatascience.com/denoising-autoencoders-explained-dbb82467fc2>
    - Generate data with AE: https://towardsdatascience.com/how-to-generate-new-data-in-machine-learning-with-vae-variational-autoencoder-applied-to-mnist-ca68591acdcf
    - Use case: <http://www.opendeep.org/v0.0.5/docs/tutorial-your-first-model>
  + Explanation 2
    - <https://www.hindawi.com/journals/sp/2017/3610378/>
    - Denoising Autoencoder (DAE)
    - Generally, the structure of AE [27] is shown in Figure 1. Here, the whole system consists of two networks, that is, encoder and decoder. Its purpose is to make the reconstruction layer output as similar to the input as possible. The coding network will code and calculate the input and then reconstruct the result to by the decoder. And denoising automatic coding is developed according to the automatic coding, it will learn a more robust representation of the input signal and has stronger generalization ability than ordinary encoders by adding noise to the training data.
* Tesseract Training
  + <https://github.com/tesseract-ocr/tessdoc/blob/40c456ce6a479fc30cb519f9886ae433c0dff17e/TrainingTesseract-4.00.md>
* RNN, LSTM architecture
  + <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>
* RNN for spelling correction mechanism
  + <https://lionbridge.ai/articles/difference-between-cnn-and-rnn/#:~:text=The%20main%20difference%20between%20CNN,as%20a%20sentence%20for%20example.&text=Whereas%2C%20RNNs%20reuse%20activation%20functions,next%20output%20in%20a%20series>.
* 1000 Faster, non deep learning
  + <https://towardsdatascience.com/symspellcompound-10ec8f467c9b>
  + Symmetric Deleting Method
    - 1) get a dictionary of words
    - 2) compute an expanded dictionary, based on the dictionary and a given edit distance. Len(dict) for a particular word of length N is N.
      * E.g. for “eye” and edit distance = 1, we have “ye”, “ee”, “ey”.
    - 3) For the given word, search the direct match in the expanded dictionary, rather than the word itself. This is computationally less expensive.
* OCR Errors
  + Survey of typical OCR errors
  + \*Impact of OCR errors on paragraph, context recognition
    - <http://www.cse.lehigh.edu/~lopresti/tmp/AND08journal.pdf>
* Census Linking
  + <https://usa.ipums.org/usa/mlp_linking_method.shtml>

<https://www.ijcaonline.org/archives/volume143/number4/kumar-2016-ijca-910142.pdf>

# Academic

AHA Annual Meeting 2022, deadline 2021, February.

historians.org/about-aha-and-membership/annual-meeting/future-meetings

# Data Source Related Program

USPS data source

<https://about.usps.com/publications/pub119.pdf> (introduction to all sources of information about USPS in the 19th century).

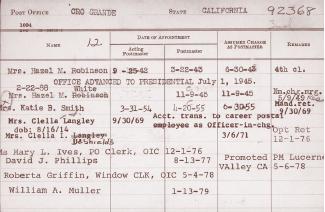
<https://about.usps.com/publications/pub119/pub119_v03_026.htm#:~:text=The%20Record%20of%20Appointment%20of,are%20given%20beginning%20in%201824>. (general intro. )

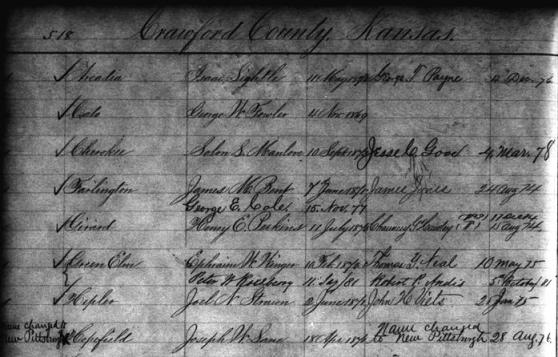
<https://www.amazon.com/-/es/Richard-W-Helbock-ebook/dp/B0032FPVS6/ref=sr_1_2?dchild=1&ie=UTF8&keywords=United%20States%20Post%20Offices%20Richard%20Helbock&language=en_US&qid=1605451229&sr=8-2> (Richard-W-Helbock listing of post offices, data source.)

Record of Appointment of Postmasters, 1814–1971

“ The records show the names of Post Offices, the dates of their establishment and discontinuance, any name changes, and the names and appointment dates of postmasters. ”

Collected from appointment ledgers for postmasters.

 (1900s)



我在做的数据：美国central gov注册的所有雇员。数据包括邮递员的名字，工作邮局地点，可以推算邮局地址和邮局收入。 他的数据：1）主要差别是：没有任何收入信息；并且没有雇员姓名；局限于西部。2）而他的出版物使用的是别人整理的另外一个数据库，非中央政府的注册。“ 邮局部门”每雇佣一名雇员都有一个“注册簿”，和中央政府雇员分开统计。类似于打卡。收集这些注册簿的人因为技术问题似乎只收集了邮局地点和位置的变化。没有雇员的信息。

# Others

Visualize Software (UML diagrams)

<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/>

<https://stackoverflow.com/questions/260165/whats-the-best-way-to-generate-a-uml-diagram-from-python-source-code>

Dataset for handwritten digits and characters

<https://www.nist.gov/srd/nist-special-database-19> (NIST)

<https://medium.com/overfitted-microservices/cracking-the-handwritten-digits-recognition-problem-with-scikit-learn-b5afc28e2c24> (MNIST)

Synthetic Noise

(synthetic data) <https://www.semanticscholar.org/paper/Generation-of-training-database-using-a-noise-model-Yen-Kim/29dc5f19a3cda1b49d588fb5630bf73172952ed9>

(model noise) <http://kmdouglass.github.io/posts/modeling-noise-for-image-simulations/>

(synthetic data) <https://github.com/search?q=synthetic+data+for+ocr>

(synthetic data generator) https://github.com/gofortargets/sythetic\_data\_ocr

(custom trained pytesseract traindata) <http://wordcapturex.deskperience.com/deployment/wordcapturex-ocr-languages>