NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

Project Report 2 :

Text Image Segmentation for Optimal OCR (Optical character recognition)

CZ4003 – COMPUTER VISION
(Semester 1, AY 2020/2021)

WILSON THURMAN TENG
U1820540H

Table of Contents

1. Introduction

1.1 Overview

In this report, various algorithms are implemented to improve the accuracy of optical character recognition (OCR) on 2 images, “sample01.png” and “sample02.png. OCR aims to recognize texts in imaged documents and is one of the earliest computer vision techniques that have been commercialized successfully. An open-source OCR software, Tesseract is used.

OCR usually involves a series of image processing and recognition tasks including:

1. Text image binarization to convert a colour/grayscale image into a binary image with multiple foreground regions (usually characters).
2. Connected component labelling to detects each binarized character region.
3. Character recognition by using classifiers (E.g. pre-trained neural network).

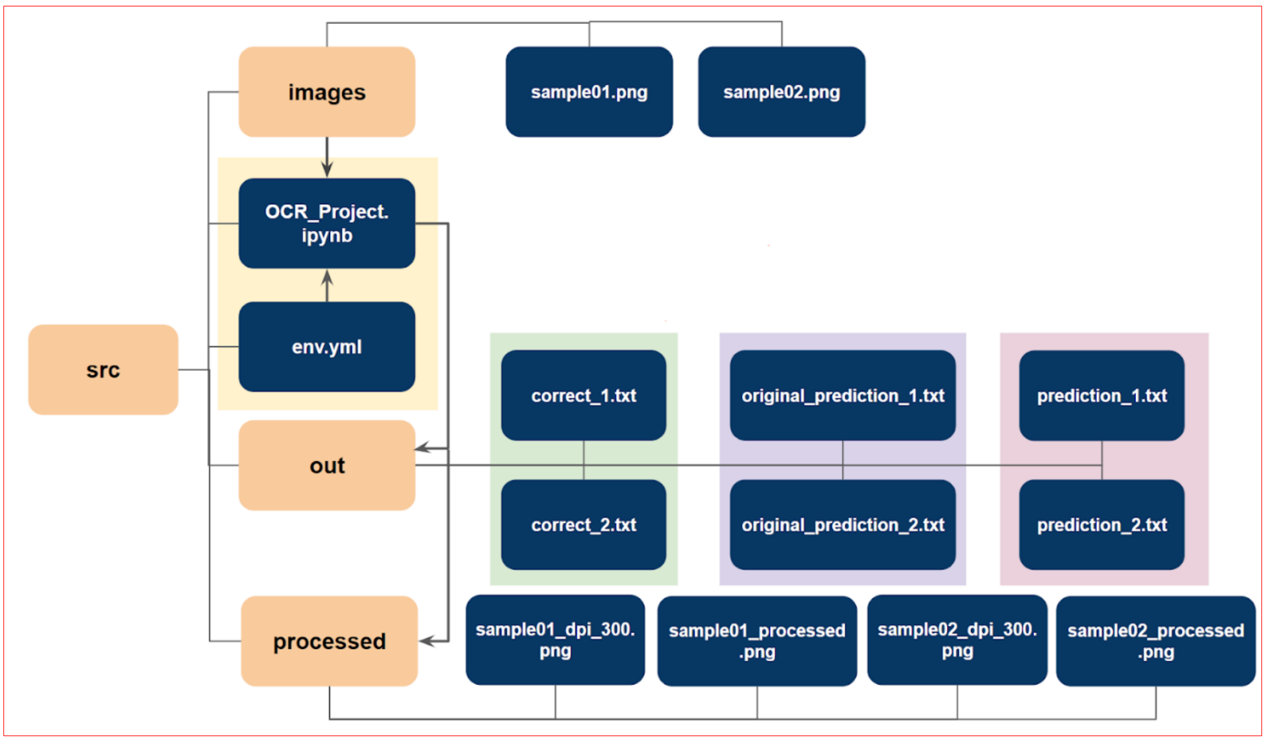
1.2 Dependencies

The following tools will be used in the implementations of this lab report:

- Python v3.6.11 (Source code tested on this version)
- Conda v4.9.2
    - Dependencies listed in ‘src/env.yml’
- Jupyter Notebook
- Tesseract v5.0.0-alpha.20200328

1.3 Program Structure

The files in the source code are arranged as follows:



- **Images** contain the original images.
- **Out** contains the text files of the images (ground truth text, original prediction, prediction after processing).
- **Processed** contains the processed images by *'OCR_Project.ipynb'*.
- *OCR_Project.ipynb'* is the script which contains all the processing code'
- *'env.yml'* is the yaml dependency file for Anaconda.

1.4 Tesseract Setup

```
custom_config = f'-c tessedit_char_whitelist="{whitelist}" --psm 3 --oem 0'
```

The configuration above is defined for the Tesseract engine to ensure that the results in this report are replicable.

```
Whitelist: abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ:-.,!\"\'
```

A whitelist of the above characters are also used to ensure that the prediction does not introduce any unwanted special characters.

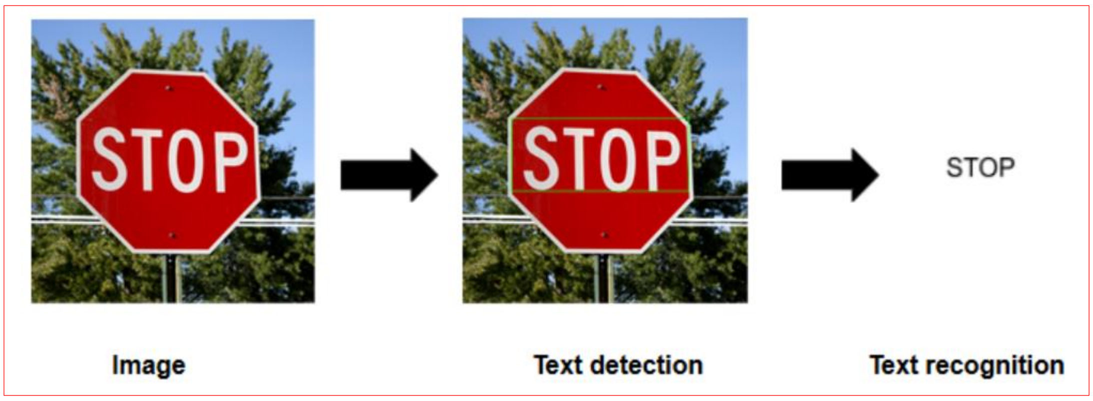
```
0 = Original Tesseract only.
1 = Neural nets LSTM only.
2 = Tesseract + LSTM.
3 = Default, based on what is available.
```

OEM refers to the OCR Engine Mode. In this setup, 0 is chosen which uses the legacy version of the Tesseract engine.

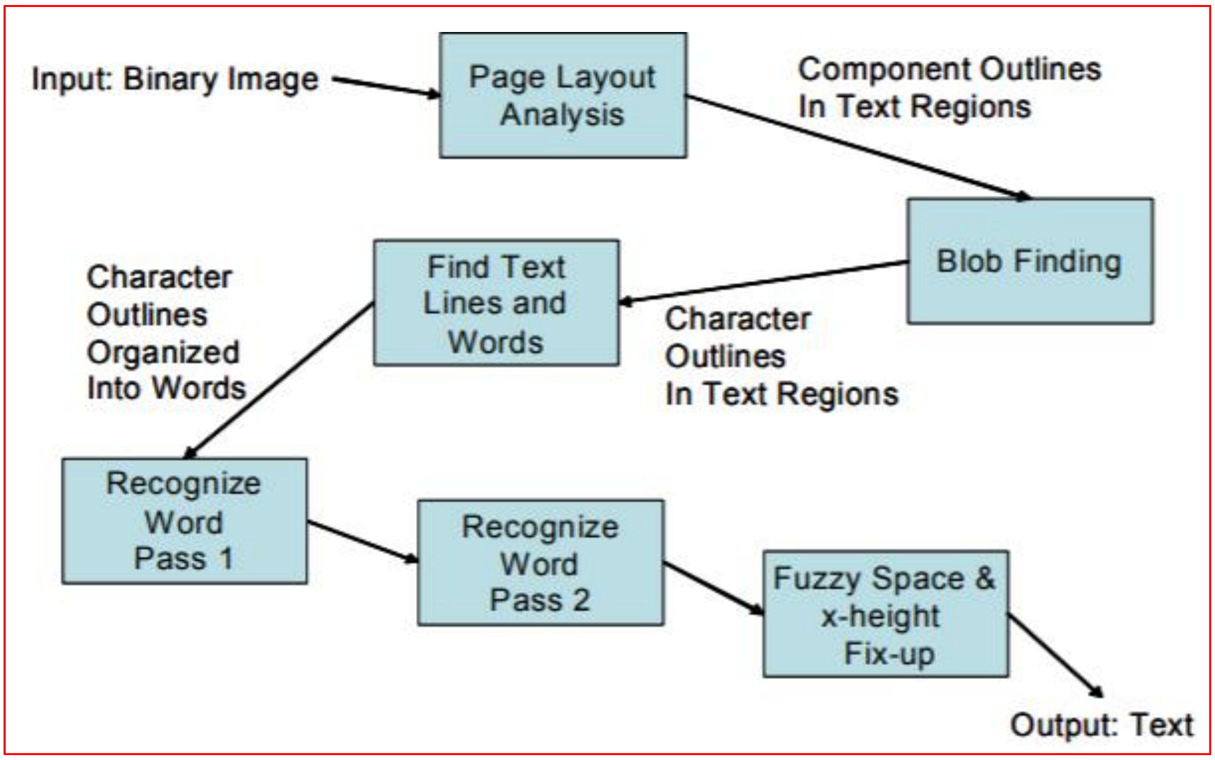
```
0 = Orientation and script detection (OSD) only.
1 = Automatic page segmentation with OSD.
2 = Automatic page segmentation, but no OSD, or OCR. (not implemented)
3 = Fully automatic page segmentation, but no OSD. (Default)
4 = Assume a single column of text of variable sizes.
5 = Assume a single 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.
```

PSM refers to the Page Segmentation Mode. In this setup, the default mode 3 is selected. This mode features automatic page segmentation but there will be no orientation and script detection.

2. How does Tesseract OCR work?



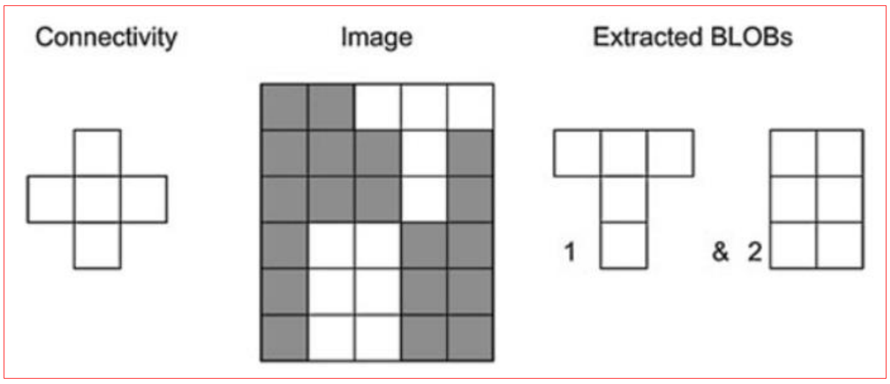
Optical character recognition, widely known as OCR, is an open-source technology developed for converting text from a document or image into editable and searchable text.


Input: Binary Image
Page Layout Analysis
Component Outlines In Text Regions
Blob Finding
Character Outlines In Text Regions
Find Text Lines and Words
Character Outlines Organized Into Words
Recognize Word Pass 1
Recognize Word Pass 2
Fuzzy Space & x-height Fix-up
Output: Text

The main stages of Tesseract OCR are as follows:

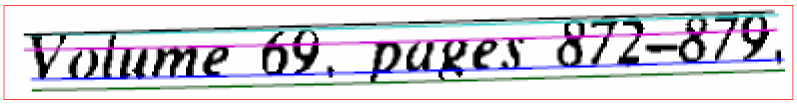
1. Connected Component Analysis
2. Text Line Segmentation
3. Word Segmentation
4. Word Recognition

Connected Component Analysis



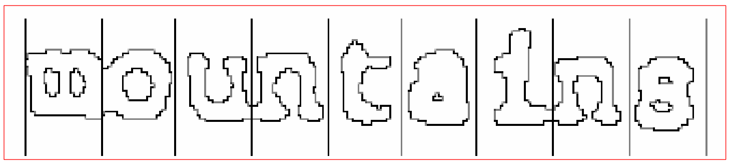
The first step is a connected component analysis where the outlines of a component in a given image are uniquely identified. As shown below, the outlines of the white blocks are gathered/nested together. Afterwards, they are extracted into Blobs (Binary Large Object).

Text Line Segmentation



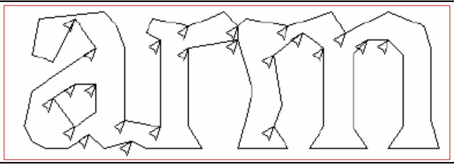
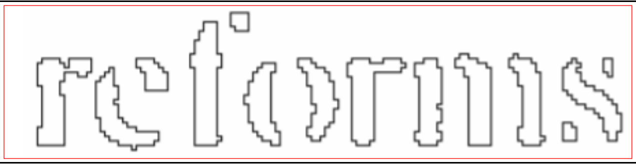
Blobs are organized into text lines and a baseline is established. This is done through partitioning the blobs into groups with a reasonably continuous displacement for the original straight baseline.

Word Segmentation



Afterwards, the lines and regions are analysed for **fixed pitch** or **proportional text**. When a fixed pitch is detected, it is likely that there is a kind of character spacing. In this case, the word is then segmented or chopped off into characters.

Word Recognition

	
With chop points	After chops

After which, Tesseract identifies concave vertices of a polygonal approximation of the outline and uses chop points to separate the joined characters. The chop is only executed if there is an improve in the confidence of the result.

Each word that is satisfactory is passed to an adaptive classifier as training data. The adaptive classifier then gets a chance to more accurately recognize text lower down the page.

3. Metrics

The similarity metrics used in this report lean heavily towards those used in Natural Language Processing (NLP), which is suitable for this task.

In this project, the following metrics are used:

1. Jaccard Similarity
2. Cosine Similarity
3. Levenshtein Similarity

The 3 metrics are then averaged, and the mean score is used to compare the performance of the pre-processing performed.

Source code:

```python
custom_config = f'-c tessedit_char_whitelist="{whitelist}" --psm 3 --oem 0'

def get_metric_score(img, ground_truth_pth):
    img_str = pytesseract.image_to_string(img, config=custom_config).strip()
    print(f"{'=' * 18} Results {'=' * 18}")
    print(f"\n{indent(img_str, 3)}")
    with open(ground_truth_pth, 'r', encoding="utf8") as f:
        ground_truth = f.read().strip()

        jaccard_sim = get_jaccard_sim(img_str, ground_truth)
        cosine_sim = get_cosine_sim(img_str, ground_truth)
        levenshtein_sim = get_levenshtein_sim(img_str, ground_truth)

        print(f"-> Jaccard Similarity:    \t{jaccard_sim:.5f}")
        print(f"-> Cosine Similarity:     \t{cosine_sim:.5f}")
        print(f"-> Levenshtein Similarity:\t{levenshtein_sim:.5f}")

        avg_score = (jaccard_sim + cosine_sim + levenshtein_sim) / 3
        print(f"\n-> Average Score: {avg_score:.5f}")

    return avg_score, [jaccard_sim, cosine_sim, levenshtein_sim], img_str
```

3.1 Jaccard Similarity

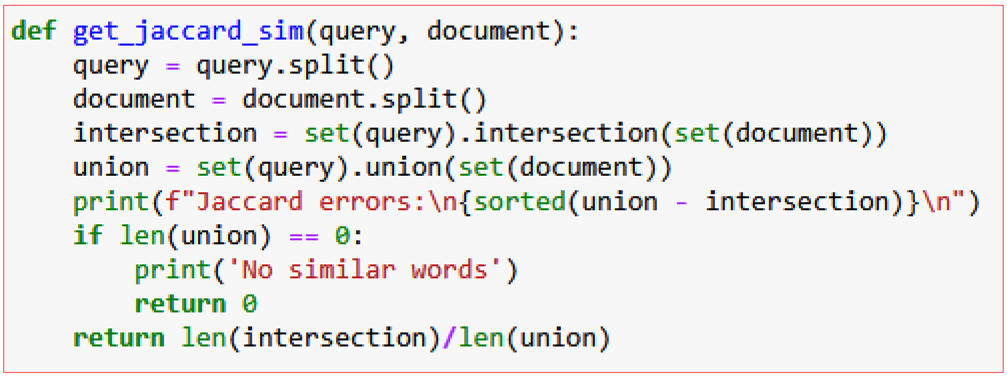
$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Jaccard similarity computes the percentage of common words between 2 strings over all possible words from both strings.

Jaccard Similarity is used in NLP to find out the similarity between 2 strings by the words used in those 2 strings. Punctuation and stop words (i.e. Common words used in the language, pronouns are an example of stop words) are often removed before calculating the Jaccard similarity. Words are also converted to their base form (i.e. past tense word ‘done’ is changed to ‘do’).

However, in this project, since we are also interested in converting punctuation accurately, we split the passage by whitespaces and use these tokens to compute the Jaccard similarity.

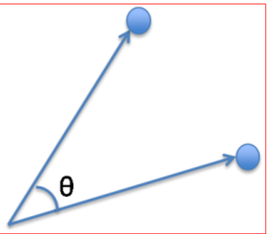
Source Code:



```
def get_jaccard_sim(query, document):
    query = query.split()
    document = document.split()
    intersection = set(query).intersection(set(document))
    union = set(query).union(set(document))
    print(f"Jaccard errors:\n{sorted(union - intersection)}\n")
    if len(union) == 0:
        print('No similar words')
        return 0
    return len(intersection)/len(union)
```

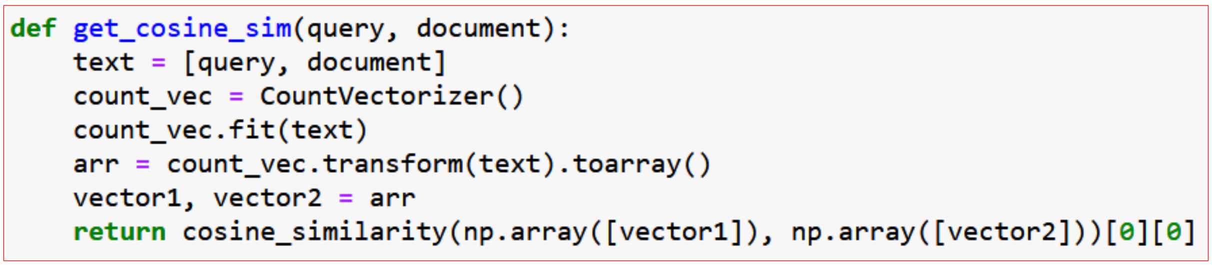
3.2 Cosine Similarity

$$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$



The cosine similarity assumes both strings can be represented in the vector space and takes the normalized dot product of these 2 vectors.

Source Code:

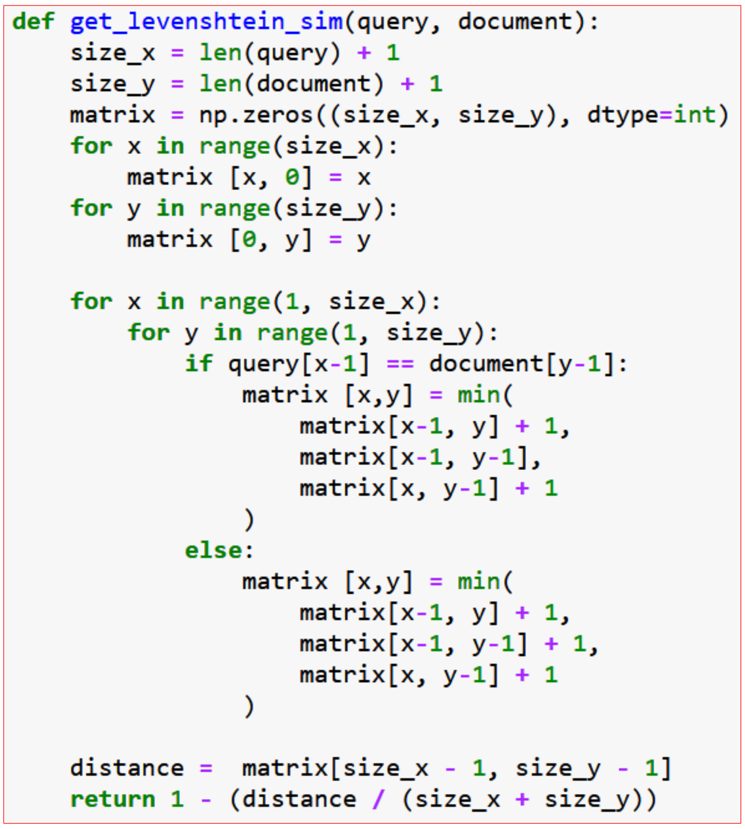


```
def get_cosine_sim(query, document):
    text = [query, document]
    count_vec = CountVectorizer()
    count_vec.fit(text)
    arr = count_vec.transform(text).toarray()
    vector1, vector2 = arr
    return cosine_similarity(np.array([vector1]), np.array([vector2]))[0][0]
```

3.3 Levenshtein Similarity

To compute the Levenshtein similarity between 2 strings, we compute the amount of insertion, deletion or substitution operations necessary to convert 1 string into the other. The amount of change the string needs to undergo is then computed as the percentage of changes over the combined length of both strings. Levenshtein similarity is then computed by subtracting this percentage from 1.

Source Code:



```python
def get_levenshtein_sim(query, document):
    size_x = len(query) + 1
    size_y = len(document) + 1
    matrix = np.zeros((size_x, size_y), dtype=int)
    for x in range(size_x):
        matrix [x, 0] = x
    for y in range(size_y):
        matrix [0, y] = y

    for x in range(1, size_x):
        for y in range(1, size_y):
            if query[x-1] == document[y-1]:
                matrix [x,y] = min(
                    matrix[x-1, y] + 1,
                    matrix[x-1, y-1],
                    matrix[x, y-1] + 1
                )
            else:
                matrix [x,y] = min(
                    matrix[x-1, y] + 1,
                    matrix[x-1, y-1] + 1,
                    matrix[x, y-1] + 1
                )

    distance =  matrix[size_x - 1, size_y - 1]
    return 1 - (distance / (size_x + size_y))
```

4. Algorithms

4.1 Otsu algorithm

Otsu is used to define a globally optimal threshold under the assumption that the intensity distribution is bimodal. An exhaustive search to minimize intra-class variance is performed on all possible threshold values. The minimization results in the maximization of inter-class variance which allows us to arrive at the desired threshold.

$$\boxed{\sigma^2(L_i) = w_a\sigma_a^2(L_i) + w_b\sigma_b^2(L_i)}$$

L_i corresponds to the intensity that is tested.
w_a and **w_b** corresponds to the probabilities of the two classes separated by a threshold. This is computed as the percentage of pixels in a class.
σ_a and **σ_b** corresponds to the standard deviation of the two classes.

$$\boxed{L^* = \arg\min_{L_i}[\sigma^2(L_i)]}$$

L^* corresponds to the globally optimal threshold which is computed as the intensity value which minimizes intra-class variance.

Source code:

```python
def otsu_thresholding(img):
    matrix = np.product(img.shape)
    variance_list = []
    hist, bin_edges = np.histogram(img, bins=256, range=(0,256))

    for temp_intensity in np.arange(1, 255, 1):
        thres = thresholding(img, temp_intensity)

        # Computing Weights
        w_a = np.sum(hist[:temp_intensity]) / float(matrix)
        w_b = np.sum(hist[temp_intensity:]) / float(matrix)

        # Computing Variances
        sigma_a = np.var(img[np.where(thres == 0)])
        sigma_b = np.var(img[np.where(thres == 1)])

        # Appending variance of Li to min_var list
        variance_list.append(w_a * sigma_a + w_b * sigma_b)

    L_optimal = np.nanargmin(variance_list)
    otsu_img = thresholding(img, L_optimal)

    return otsu_img, otsu_thres
```

4.2 Skew angle algorithm

This algorithm aims to correct the skew angle of the image under the assumption that the text is written in paragraphs which will result in a rectangular bounding box.

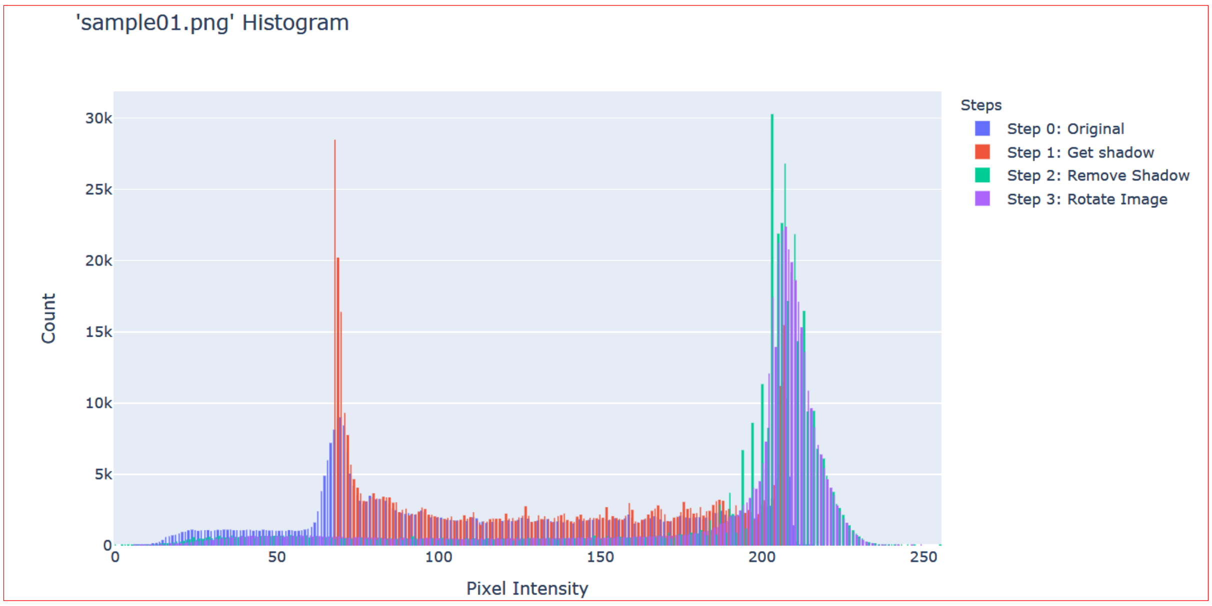
This is done by first performing Otsu thresholding to the image. Coordinates of the pixel of characters (intensity == 0) are then extracted. A minimum area rectangle is computed from these coordinates, and we obtain the skew angle of this computed rectangle.

This method is suitable for images where intensities of the characters and its background are easily separable which allows the computed rectangle to be accurate to the passage.

Source Code:

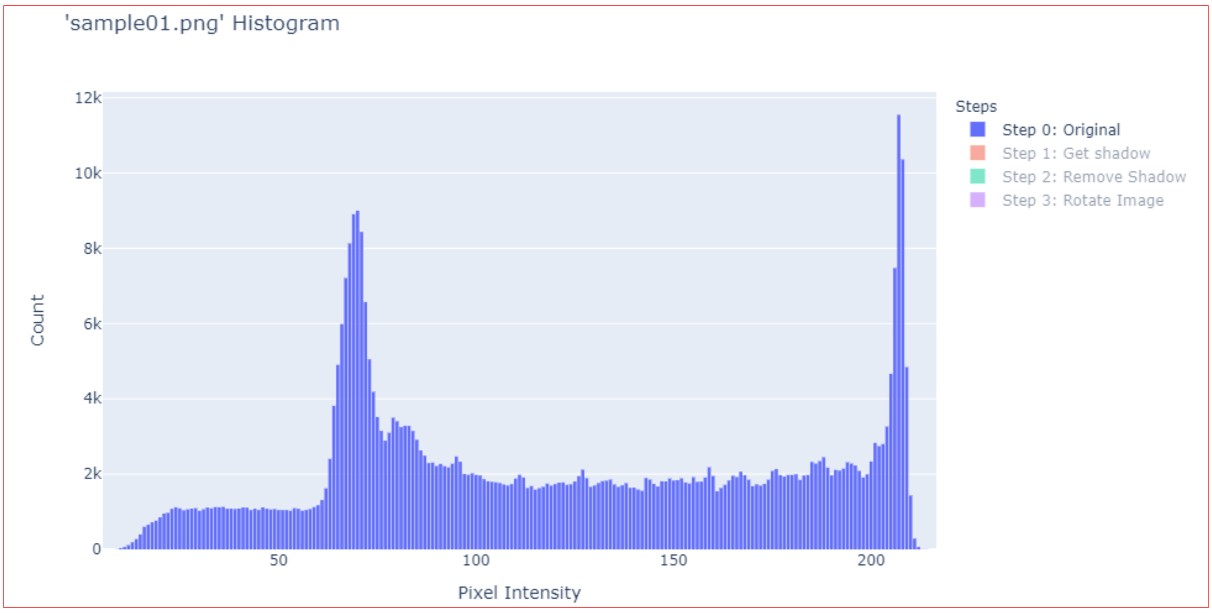
```python
def get_skew_angle(img):
    otsu, otsu_thres = otsu_thresholding(img)

    coords = np.column_stack(np.where(otsu == 0))
    angle = cv2.minAreaRect(coords)[-1]

    if angle < -45:
        return -(90 + angle)
    else:
        return -angle
```

5. Pre-processing



The histograms used in this section are from ‘sample01.png’. For an interactive experience with these histogram plots, either ‘OCR_Project.ipynb’ or ‘OCR_Project.html’ can be explored.

5.1 Step 0: Image Upscaling

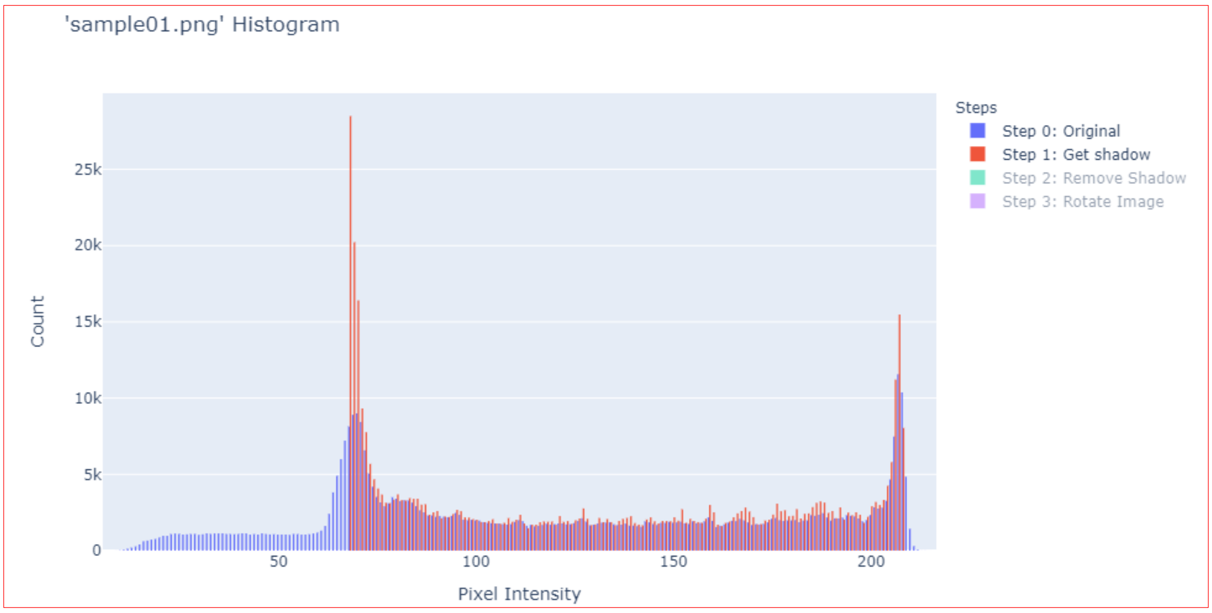


The image is first upscaled between 1.4-1.5x. This is to ensure the image has at least 300 dpi as recommended by Tesseract.

The benefits to upscaling are two-fold:

- This allows for subsequent algorithms to use larger kernels which introduces more flexibility to the shape of kernels that can be used.
- Interpolation between old pixels may create new pixels that fill in gaps between characters.

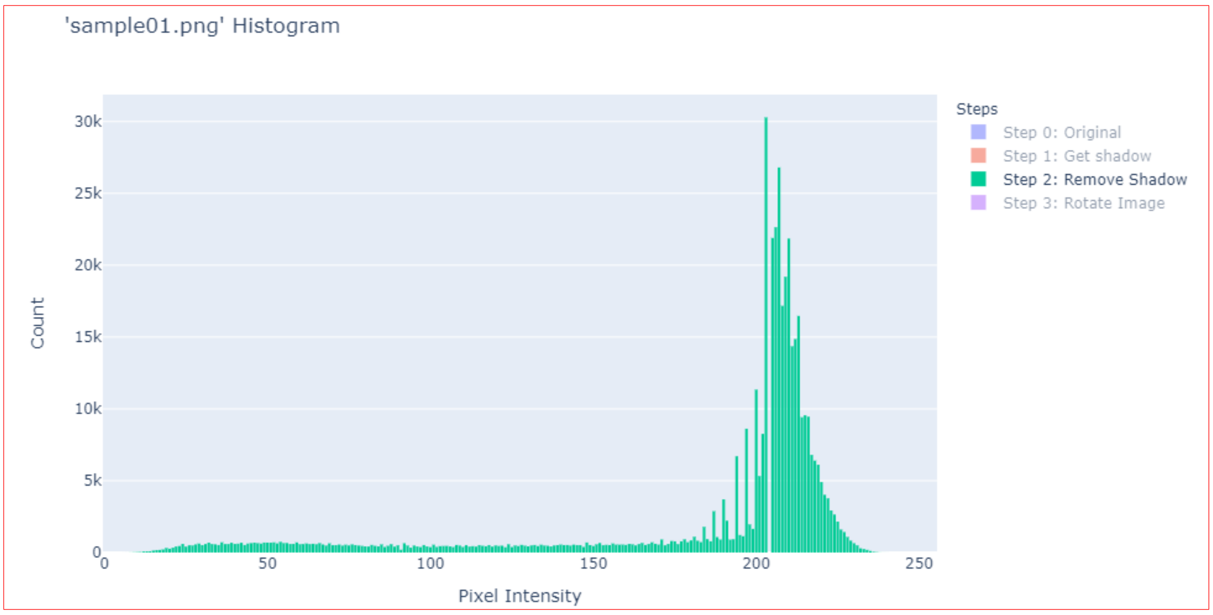
5.2 Step 1: Obtain the illumination and shadow of the image



This is achieved by applying Median Blur with a large kernel size to the image. This allows the resultant image to ignore the optical characters and produce the gradient of the image, which explains the illumination and shadow of the image well.

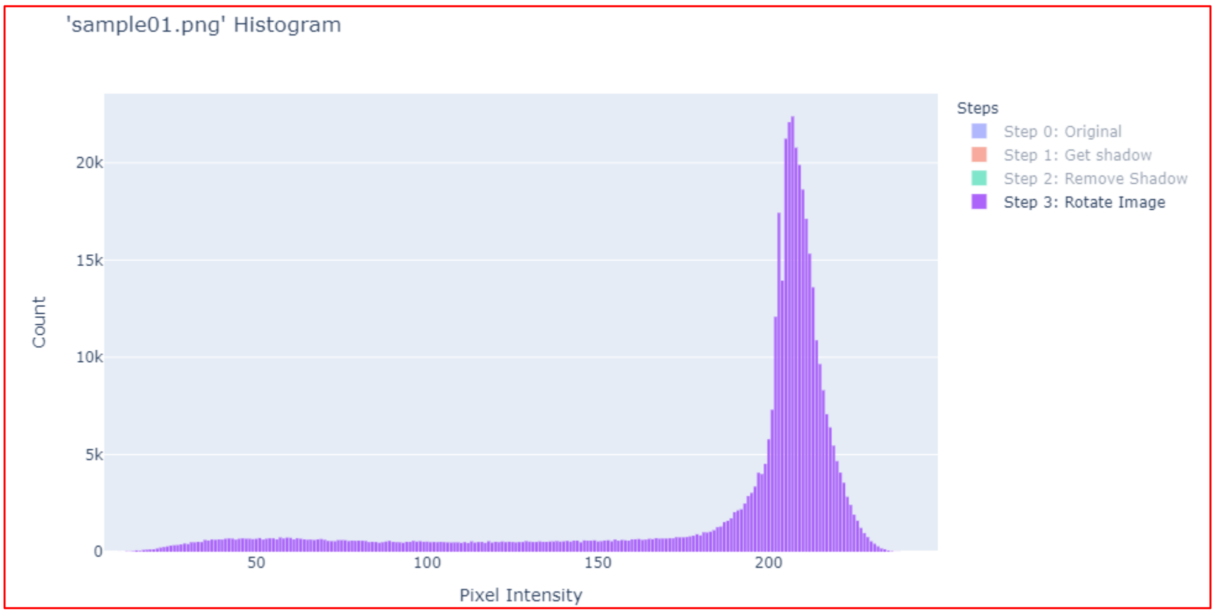
It can be observed from the histogram plot above that the shadow and the original image overlap for many intensities, hence we can conclude that a global threshold technique would be insufficient to identify an optimal threshold.

5.3 Step2: Remove illumination/shadow from image



This step reduces the effect of illumination and shadows. The original image is divided by the shadow obtained in the previous step and contrast stretched to the original [0,255] range. This darkens the previously illuminated parts of the image, while the dark parts of the image remain at similar intensities.

5.4 Step 3: Rotate image (if required)



Finally, in the last stage of pre-processing, we utilize the skew angle algorithm explained in *section 4.2* and apply an affine transformation to our original image.

It is interesting to note that the histogram of the rotated image follows a much smoother curve as compared to before rotation.

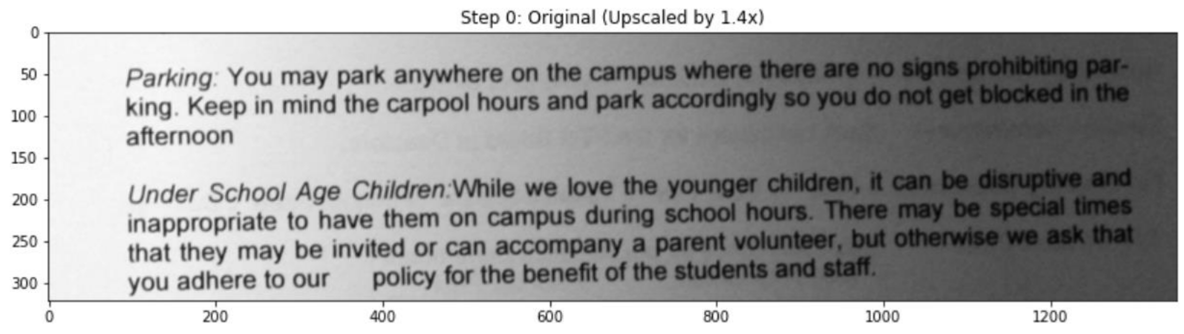
5.4 Conclusion

It can be observed in that the final histogram obtained is not bimodal, therefore binarizing the image may not be useful at all for obtaining accurate OCR results.

6. Results

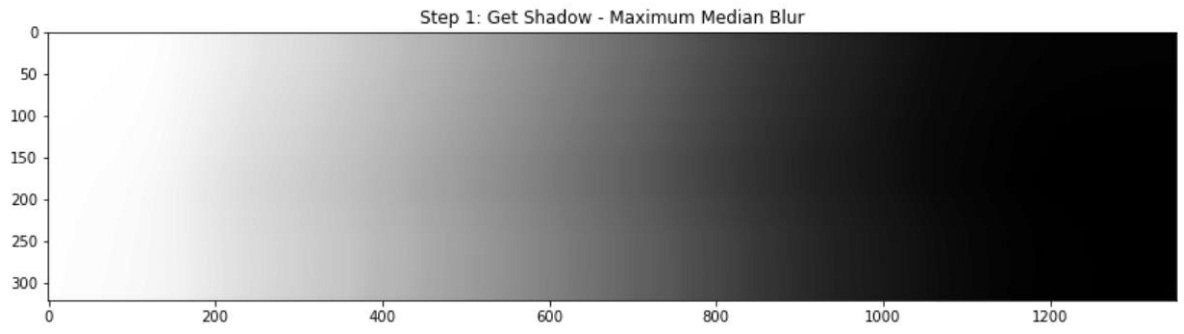
6.1 ‘sample01.png’ results (Best score achieved – 100%)

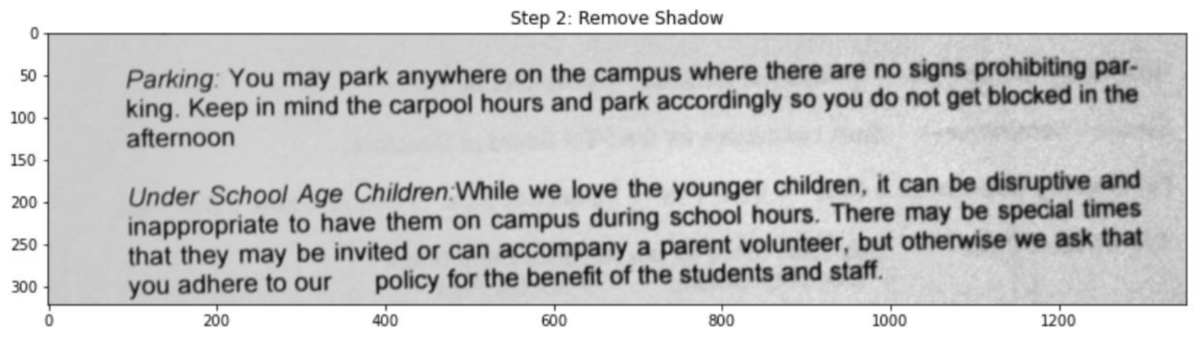
The following show the results of the pre-processing steps outlined in the previous section as well as the score achieved.


Step 0: Original (Upscaled by 1.4x)
0
50
100
150
200
250
300
0
200
400
600
800
1000
1200

Parking: You may park anywhere on the campus where there are no signs prohibiting parking. Keep in mind the carpool hours and park accordingly so you do not get blocked in the afternoon

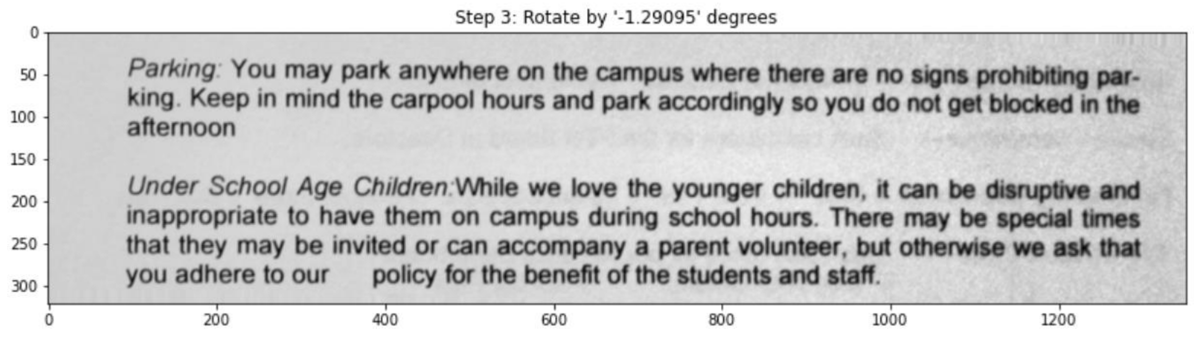
Under School Age Children: While we love the younger children, it can be disruptive and inappropriate to have them on campus during school hours. There may be special times that they may be invited or can accompany a parent volunteer, but otherwise we ask that you adhere to our policy for the benefit of the students and staff.




Step 2: Remove Shadow
0
50
100
150
200
250
300
0
200
400
600
800
1000
1200

Parking: You may park anywhere on the campus where there are no signs prohibiting parking. Keep in mind the carpool hours and park accordingly so you do not get blocked in the afternoon

Under School Age Children: While we love the younger children, it can be disruptive and inappropriate to have them on campus during school hours. There may be special times that they may be invited or can accompany a parent volunteer, but otherwise we ask that you adhere to our policy for the benefit of the students and staff.


Step 3: Rotate by '-1.29095' degrees
0
50
100
150
200
250
300
0
200
400
600
800
1000
1200

Parking: You may park anywhere on the campus where there are no signs prohibiting parking. Keep in mind the carpool hours and park accordingly so you do not get blocked in the afternoon

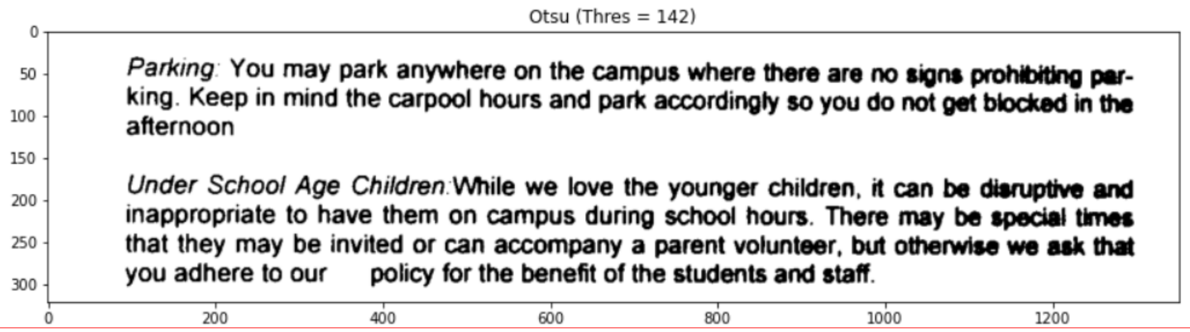
Under School Age Children: While we love the younger children, it can be disruptive and inappropriate to have them on campus during school hours. There may be special times that they may be invited or can accompany a parent volunteer, but otherwise we ask that you adhere to our policy for the benefit of the students and staff.

```
================== Results ==================

   '''
   Parking: You may park anywhere on the campus where there are no signs prohibiting par-
   king. Keep in mind the carpool hours and park accordingly so you do not get blocked in the
   afternoon

   Under School Age Children:While we love the younger children, it can be disruptive and
   inappropriate to have them on campus during school hours. There may be special times
   that they may be invited or can accompany a parent volunteer, but otherwise we ask that
   you adhere to our policy for the benefit of the students and staff.
   '''

Jaccard errors:
[]

-> Jaccard Similarity:          1.00000
-> Cosine Similarity:           1.00000
-> Levenshtein Similarity:      1.00000

-> Average Score: 1.00000
```

Experimentation with Otsu and adaptive thresholding on the processed image was also attempted but these binarization algorithms did not improve the score. This could be attributed to the unimodal distribution of the processed image observed.

6.1.1 Otsu Thresholding

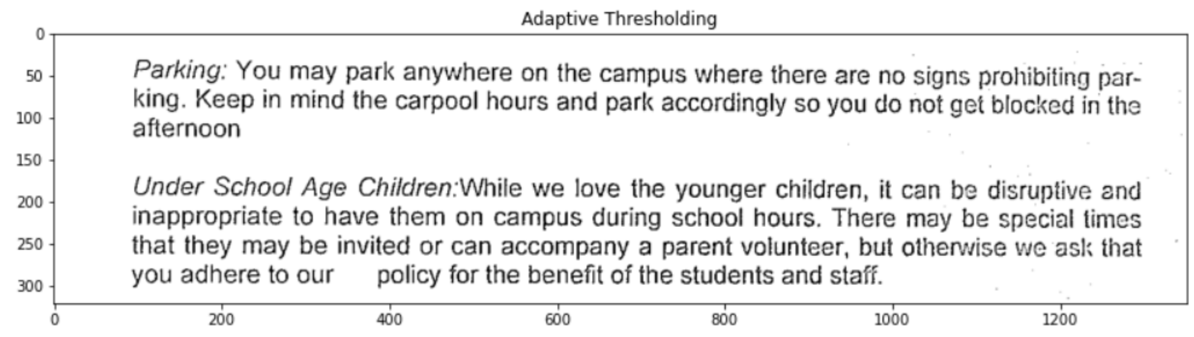
```
================== Results ==================

   '''
   Parking You may park anywhere on the campus where there are no igNs prohibiting par-
   king. Keep in mind the carpool hours and park accordingly so you do not get blocked in the
   alternoon

   Under School Age Children While we love the younger children, it can be disruptive and
   inappropriate to have them on campus during school hours. There may be special times
   that they may be invited or can accompany a parent volunteer, but otherwise we ask that
   you adhere to our policy for the benefit of the students and staff.
   '''

Jaccard errors:
['Children', 'Children:While', 'Parking', 'Parking:', 'While', 'afternoon', 'alternoon', 'igNs', 'signs']

-> Jaccard Similarity:          0.88158
-> Cosine Similarity:           0.98817
-> Levenshtein Similarity:      0.99516

-> Average Score: 0.95497
```


Otsu (Thres = 142)
0
50
100
150
200
250
300
0
200
400
600
800
1000
1200
Parking: You may park anywhere on the campus where there are no signs prohibiting par-
king. Keep in mind the carpool hours and park accordingly so you do not get blocked in the
afternoon
Under School Age Children:While we love the younger children, it can be disruptive and
inappropriate to have them on campus during school hours. There may be special times
that they may be invited or can accompany a parent volunteer, but otherwise we ask that
you adhere to our policy for the benefit of the students and staff.

It can be observed that Otsu thresholding results in thick strokes caused by the remaining slight skew. This affects the accuracy of the OCR. Eroding and dilation were also experimented with after Otsu thresholding to make the strokes thinner, but the results were not promising. The benefits of thinning the characters with wide strokes was offset by characters which are already sufficiently thin.

6.1.2 Adaptive Thresholding

```
================== Results ==================
   '''
   Parking: You may park anywhere on the campus where there are no signs prohibiting par-
   king. Keep in mind the carpool hours and park accordingly so you do not get blocked in the
   afternoon

   Under School Age Children:While we love the younger children, it can be disruptive and
   inappropriate to have them on campus during school hours. There may be special times
   that they may be invited or can accompany a parent volunteer, but otherwise we ask that
   you adhere to our policy for the benefit of the students and staff.
   '''

Jaccard errors:
[]

-> Jaccard Similarity:          1.00000
-> Cosine Similarity:           1.00000
-> Levenshtein Similarity:      1.00000

-> Average Score: 1.00000
```

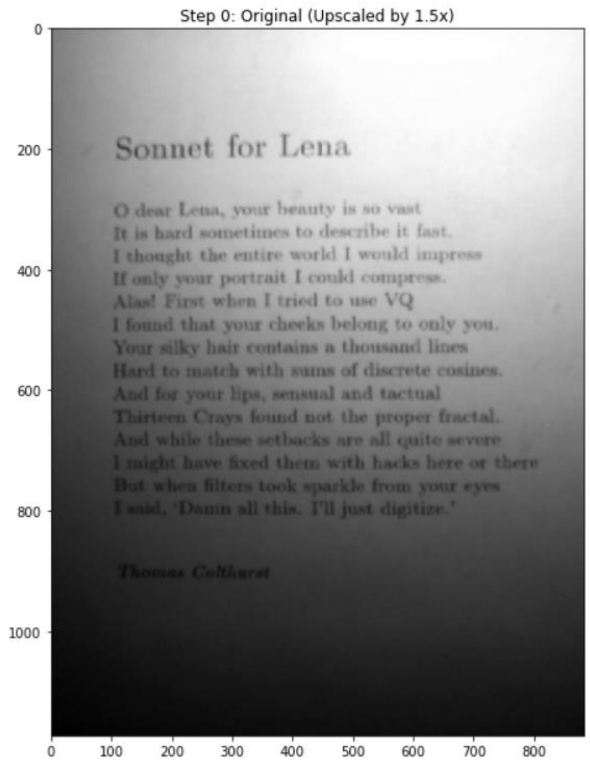

Adaptive Thresholding
0
50
100
150
200
250
300
0
200
400
600
800
1000
1200

Parking: You may park anywhere on the campus where there are no signs prohibiting par-
king. Keep in mind the carpool hours and park accordingly so you do not get blocked in the
afternoon

*Under School Age Children:*While we love the younger children, it can be disruptive and
inappropriate to have them on campus during school hours. There may be special times
that they may be invited or can accompany a parent volunteer, but otherwise we ask that
you adhere to our policy for the benefit of the students and staff.

Adaptive thresholding is also applied to the processed image. This allows for locally optimal thresholding which results in a clean binarized image.

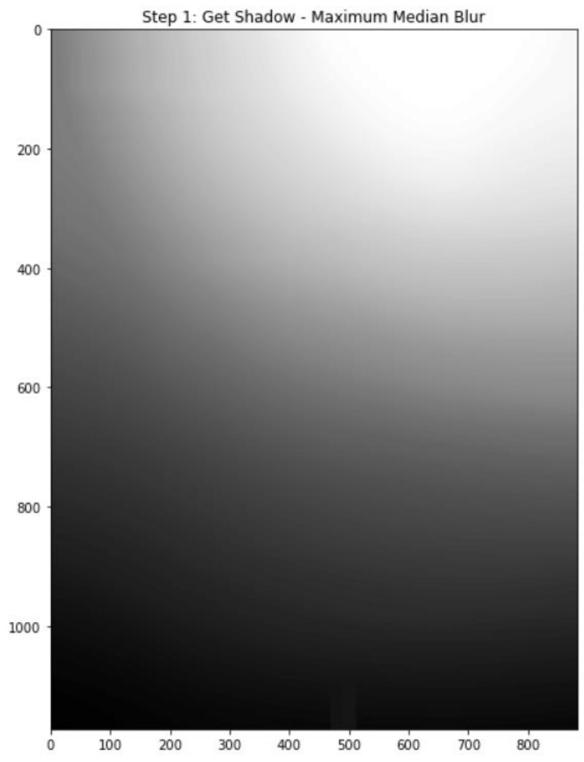
6.2 'sample02.png' results (Best score achieved – 99.04%)

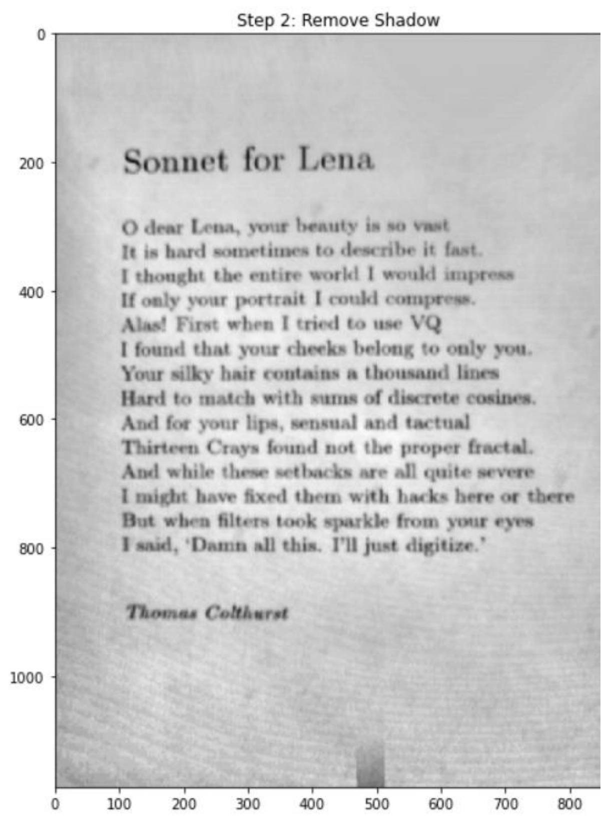

Step 0: Original (Upscaled by 1.5x)
0
200
400
600
800
1000
0
100
200
300
400
500
600
700
800

Sonnet for Lena

O dear Lena, your beauty is so vast
It is hard sometimes to describe it fast.
I thought the entire world I would impress
If only your portrait I could compress.
Alas! First when I tried to use VQ
I found that your cheeks belong to only you.
Your silky hair contains a thousand lines
Hard to match with sums of discrete cosines.
And for your lips, sensual and tactual
Thirteen Crays found not the proper fractal.
And while these setbacks are all quite severe
I might have fixed them with hacks here or there
But when filters took sparkle from your eyes
I said, 'Damn all this. I'll just digitize.'

Thomas Colthurst




Step 2: Remove Shadow
0
200
400
600
800
1000
0
100
200
300
400
500
600
700
800

Sonnet for Lena

O dear Lena, your beauty is so vast
It is hard sometimes to describe it fast.
I thought the entire world I would impress
If only your portrait I could compress.
Alas! First when I tried to use VQ
I found that your cheeks belong to only you.
Your silky hair contains a thousand lines
Hard to match with sums of discrete cosines.
And for your lips, sensual and tactual
Thirteen Crays found not the proper fractal.
And while these setbacks are all quite severe
I might have fixed them with hacks here or there
But when filters took sparkle from your eyes
I said, 'Damn all this. I'll just digitize.'

Thomas Colthurst

```
================== Results ==================

   '''
   Sonnet for Lena

   O dear Lena, your beauty is so vast

   It is hard sometimes to describe it fast.

   I thought the entire world would impress
   If only your portrait could compress.

   Alas! First when I tried to use VQ

   I found that your cheeks belong to only you.
   Your silky hair contains a thousand lines
   Hard to match with sums of discrete cosines.
   And for your lips, sensual and tactual
   Thirteen Crays found not the proper fractal.
   And while these setbacks are all quite severe
   I might have fixed them with hacks here or there
   But when filters took sparkle from your eyes
   I said, Damn all this. I'll just digitize."

   Thomas Colthurst
   '''

Jaccard errors:
["'Damn", 'Damn']

-> Jaccard Similarity:         0.97895
-> Cosine Similarity:          1.00000
-> Levenshtein Similarity:     0.99209

-> Average Score: 0.99035
```

As observed previously for ‘sample01.png’, binarization is not necessary to achieve high OCR accuracy if the processed image is clean. In this situation, only the quote before “damn” is missing from the prediction.

It was hypothesized that this “damn” quote missing could be attributed to the new Tesseract LSTM engine (oem 2) which would stipulate that the closing “ quote be accompanied by an open ‘ quote. However, after switching to the legacy engine (oem 0), the results remained the same.

Therefore, this quote missing can be attributed to the light ink in the picture.