Segmentation in OCR !!

A basic explanation of different levels of Segmentation used by the OCR system.

Susmith Reddy · Follow
Published in Towards Data Science · 8 min read · Mar 25, 2019

Welcome to ***part III*** about the working of the OCR system. In the *previous post*, we have seen the basic and widely used Preprocessing techniques in the OCR system.

In this part, we will cover the techniques of another most important phase of the OCR system, ***Segmentation.***

Segmentation is nothing but ***breaking the whole image into subparts to process them further.***

Segmentation of image is done in the following sequence :
→ Line level Segmentation
→ Word level Segmentation
→ Character level Segmentation

For all the above-mentioned levels of Segmentation, we are going to use the *Histogram Projection technique*.

Histogram Projection Method

Let me give you a brief introduction to the Histogram Projection method. Once the colored image is converted to the binary image, only black and white pixels are present in the image.

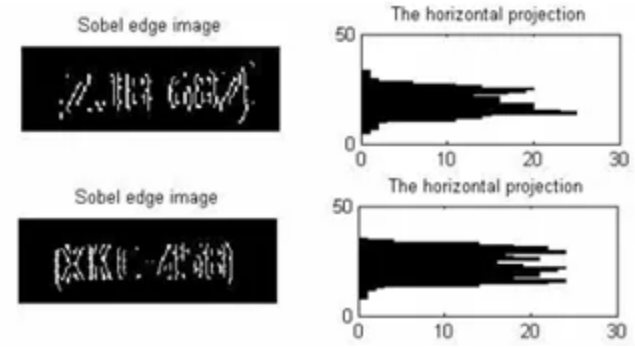
In a binary image, pixels representing useful information are called ***Foreground pixels,*** and the pixels that are not foreground pixels are called ***Background pixels.*** We choose whether a foreground pixel should be white or black while binarizing the image.

Horizontal Histogram Projection: In this method, we count the *No.of* foreground pixels along the rows of the image, and the resultant array is of the size equal to *No.of* rows in the image (Height of the image).

```
# img be binarized image of size height x width and foregound pixel
is white i.e pixel value = 255
horizontal_hist = np.sum(img,axis=1,keepdims=True)/255
# axis = 1 is used to take sum along the row
# horizontal_hist is an array of size height x 1

# If the img has foreground pixel as black i.e pixel value = 0
horizontal_hist = img.shape[1] - np.sum(img,axis=1,keepdims=True)/255
# Logic :- No.of columns - No.of white pixels
```

Plotting *horizontal_hist* as a histogram looks like this



No.of White pixels (foreground) in each row. **Source: researchgate.net**

In the above image,

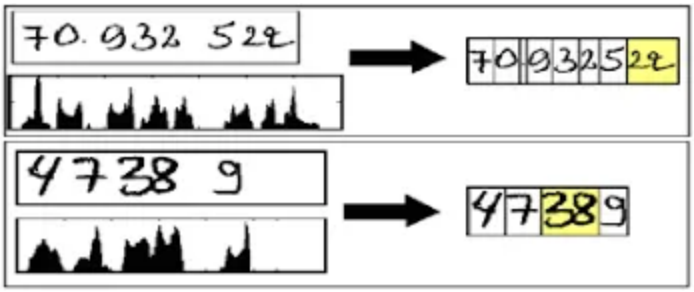
- Higher peaks imply *No.of* foreground pixels in the corresponding row is high.
- Lower peaks imply *No.of* foreground pixels in the corresponding row are low

Vertical Histogram Projection: In this method, we count the *No.of* foreground pixels along the image columns. The resultant array is of the size equal to *No.of* columns in the image (Width of the image).

```
# img be binarized image of size height x width and foregound pixel
is white i.e pixel value = 255
vertical_hist = np.sum(img,axis=0,keepdims=True)/255
# axis = 0is used to take sum along the column
# vertical_hist is an array of size 1 x width

# If the img has foreground pixel as black i.e pixel value = 0
vertical_hist = img.shape[0] - np.sum(img,axis=0,keepdims=True)/255
# Logic :- No.of rows - No.of white pixels
```

Plotting *vertical_hist* as a histogram looks like this



No.of Black pixels (foreground) in each column. **Source: researchgate.net**

Note: In the above image, foreground pixels are black, and background pixels are white.

In the above image,

- Higher peaks imply *No.of* foreground pixels in the corresponding column are high.
- Lower peaks imply *No.of* foreground pixels in the corresponding column are low.

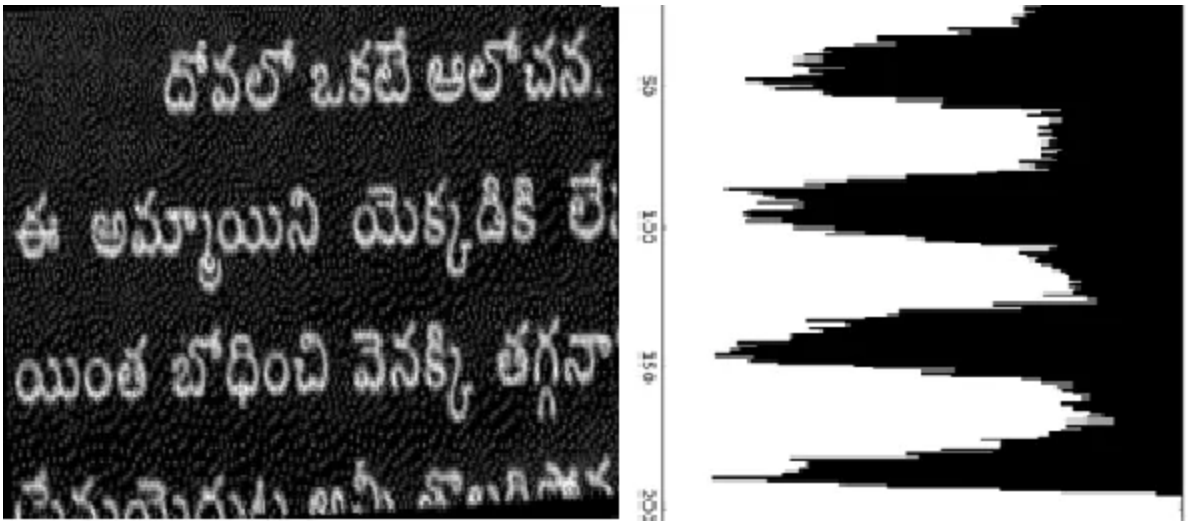
Now that we have understood *Horizontal* and *Vertical* histogram projections, we are ready to know how different levels of ***Segmentation*** is performed.

1. ***Line Level Segmentation:*** In this level of segmentation, we are provided with a *skew corrected image* containing text written in the form of lines. The objective of *Line Level Segmentation* is to ***segment the image into lines***.

> General Rules of writing: We should leave enough space in between the lines.

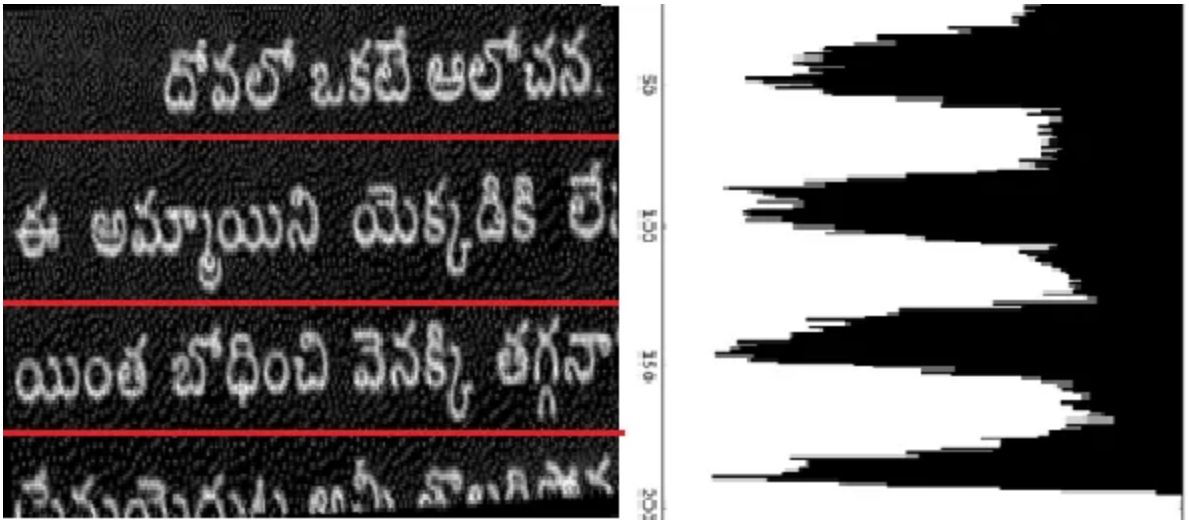
The idea is, If we *Horizontally project* the binary image,

- Rows that ***represent the text in a line have high No.of foreground pixels,*** which correspond to *higher peaks* in the histogram.
- Rows that ***represent the gaps in-between the lines have high No.of background pixels,*** which correspond to *lower peaks* in the histogram.



Horizontal Projection of image, Counting No.of Foreground pixels in each row. **Source: Image by the author.**

Rows that correspond to lower peaks in the histogram can be selected as the segmenting lines to separate the lines.



Segmenting the image as lines by selecting the rows which have lower peaks. **Source: Image by the author.**

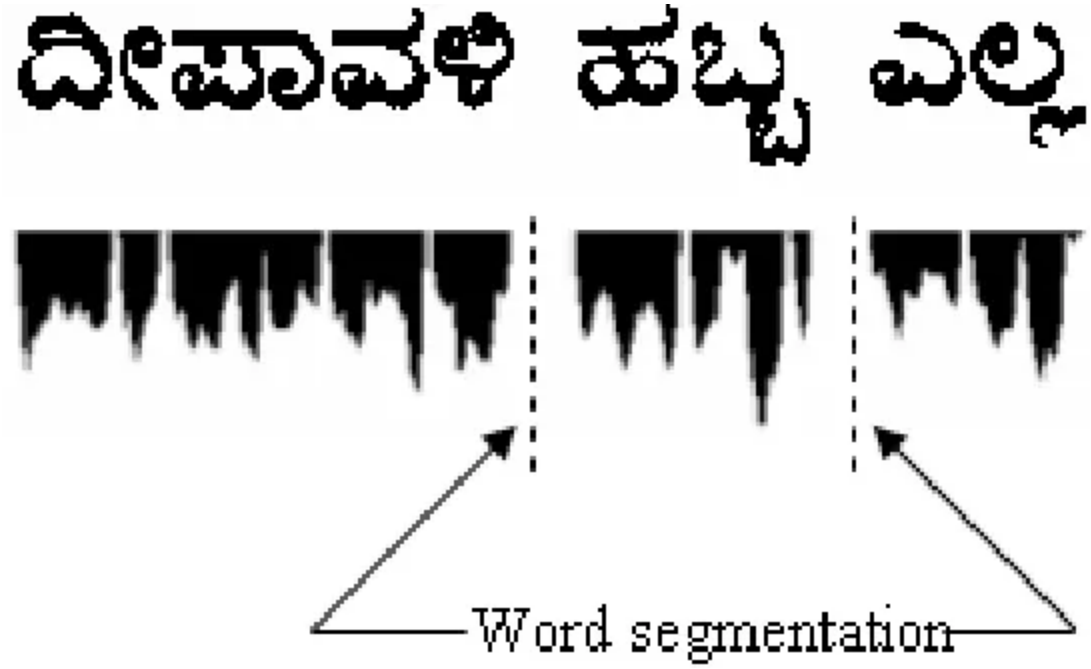
2. ***Word Level Segmentation:*** At this level of segmentation, we are provided with an image containing a single line (segmented in the previous step) which consists of a sequence of words. The objective of *Word Level Segmentation* is to ***segment the image into words.***

> General Rules of writing: We should leave enough space in between the words.

The idea is similar to the previous step, but the only change is, here we have to project the image vertically (Sum is taken along columns) because we have to segment words vertically.

If we *Vertically project* the binary image,

- Columns that ***represent the text have high No.of foreground pixels,*** which correspond to *higher peaks* in the histogram.
- Columns that ***represent the gaps in-between the words have high No.of background pixels,*** which correspond to *lower peaks* in the histogram.



Vertical Histogram Projection of image. **Source: researchgate.net**

Note: In the above pic, Foreground pixels are black pixels, and background pixels are white.

I think you have figured out how the word segmentation is done in the above picture. Columns that correspond to lower peaks in the histogram can be selected as the segmenting lines to separate the words.

> *For segmenting words, lower peaks should be selected in such a way that they should span through a certain width (threshold). This is because we'll find lower peaks which correspond to the gaps between disconnected characters within a word, which we are not interested in. As we know, the gaps between the words are greater than the gaps between the characters within a word, the threshold should be chosen in such a way that neglects the thin gaps between the characters within the words.*

3. ***Character Level Segmentation:*** At this level of segmentation, we are provided with an image containing a single word (segmented in the previous

step) which consists of a sequence of characters. The objective of *Character Level Segmentation* is to ***segment the image into individual characters.***

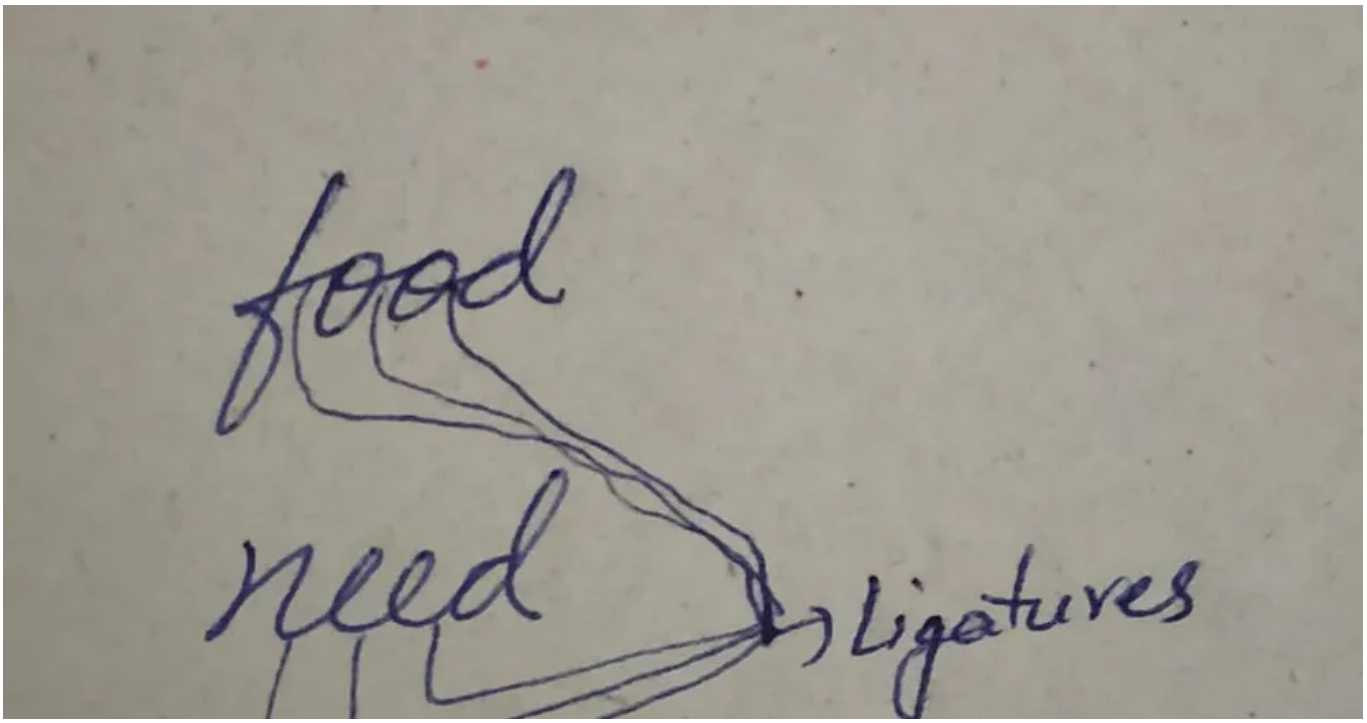
This level of segmentation is optional, which depends upon the context where the OCR is being used.

- If the OCR system is being applied to the text, where ***characters within a word are separate,*** *Character Level Segmentation* is not required. Since a uniform gap is maintained (even though it is small) between the characters within a word, we can segment the characters in the previous step itself (by setting a very low *threshold*).
- If the OCR system is being applied to the text, where ***characters within a word are joined*** (Cursive handwriting), *Character Level Segmentation* has to be performed to segment the characters.

We can use the same idea which we have used for Word Level Segmentation by leveraging the small gap between the characters, *i.e., b*y projecting the image vertically, we can segment the characters.

In *Character Level Segmentation,* segmenting of characters is not straightforward because of the concept of ***ligatures*** in Cursive handwriting.

> ***What is Ligature ??***
> *A Ligature is the link which joins the two successive characters in cursive handwriting.*



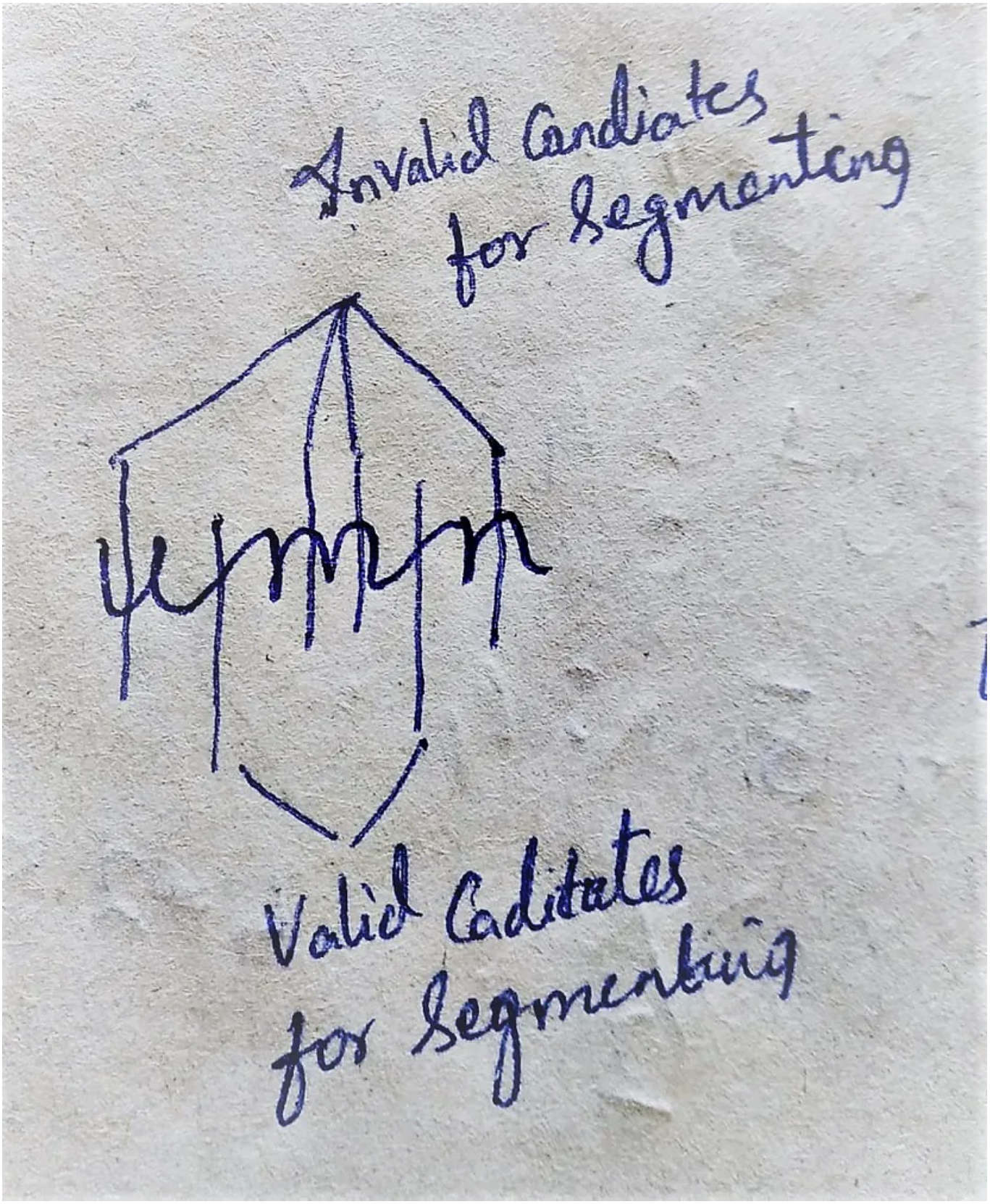
Source: Image by the author.

Now you might be thinking, ***What problem does this ligature create*** *…..right?*
In English script, we have two types of characters

- ***Closed Characters:*** These characters have a loop, and when projected vertically, all the columns spanned across these characters have at least 2 foreground pixels. ***E.g., a, b, c,d***
- ***Open Characters:*** These characters don't have a loop, and when projected vertically, Of all the columns spanned across these characters, there is at least one column that has only 1 foreground pixel. E.g., ***u, v, w***

Thus the OCR system cannot differentiate the ligature with the curve from open characters and considers both valid candidates for segmenting. This

problem is known as the ***Over Segmentation Problem.***

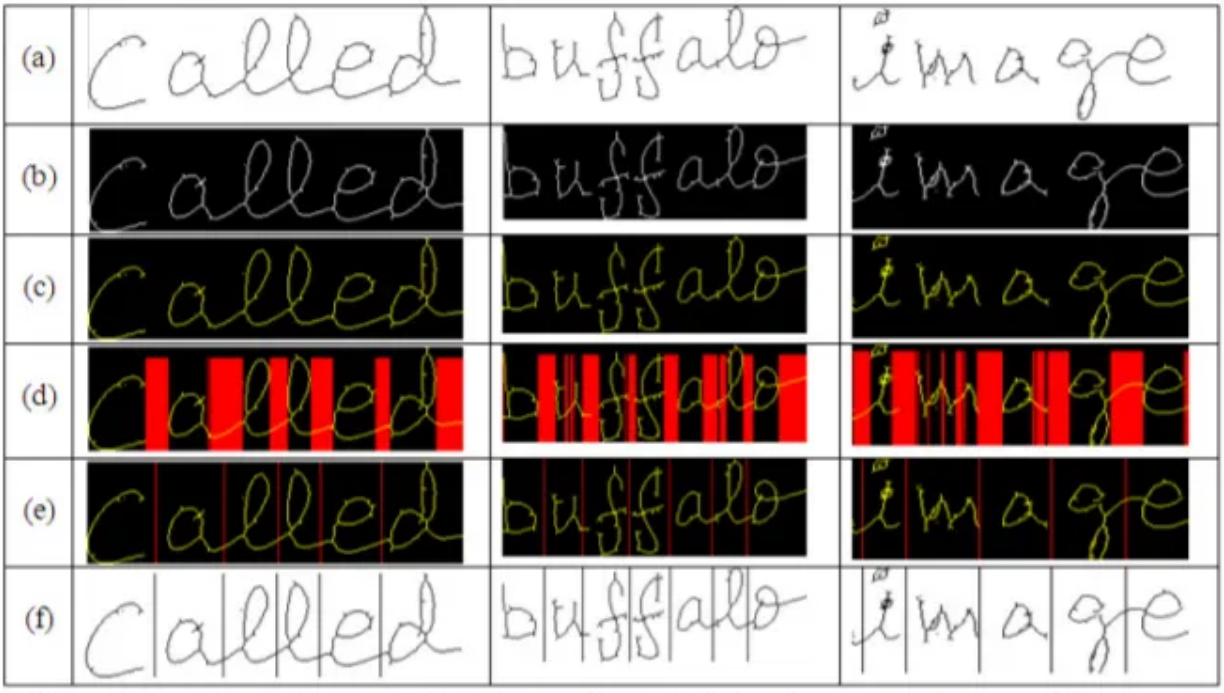


Ligature problem for the word “umn” written in Cursive. **Source: Image by the author.**

Note: In the above image, for segmenting, valid candidates are the ligatures (connections between u,m & m,n), and invalid candidates are those, which are a part of the open characters ***u, m, n*** *(we cant segment in between the character)*

Till date, we have no perfect solution for this problem, but here is the basic and widely used method to overcome this problem.

We assume ***Thinning*** is done to 1-pixel width. So possible sum in vertical histogram array is
→ ***sum=0,*** if no foreground pixel is present in the column
→ ***sum=1,*** if either ligature or part of an open character is present in the column
→ ***sum>1,*** if part of a closed character is present in the column. *E.g.,* o,p (has more than one foreground pixel in each column)



Word Image Segmentation (a) Pre-processed Word Images; (b) Inverted Binary Images; (c) RGB Images; (d) Over-segmentation in Images; (e) Image after removing Over-segmentations; (f) Final Segmented Output Word Images

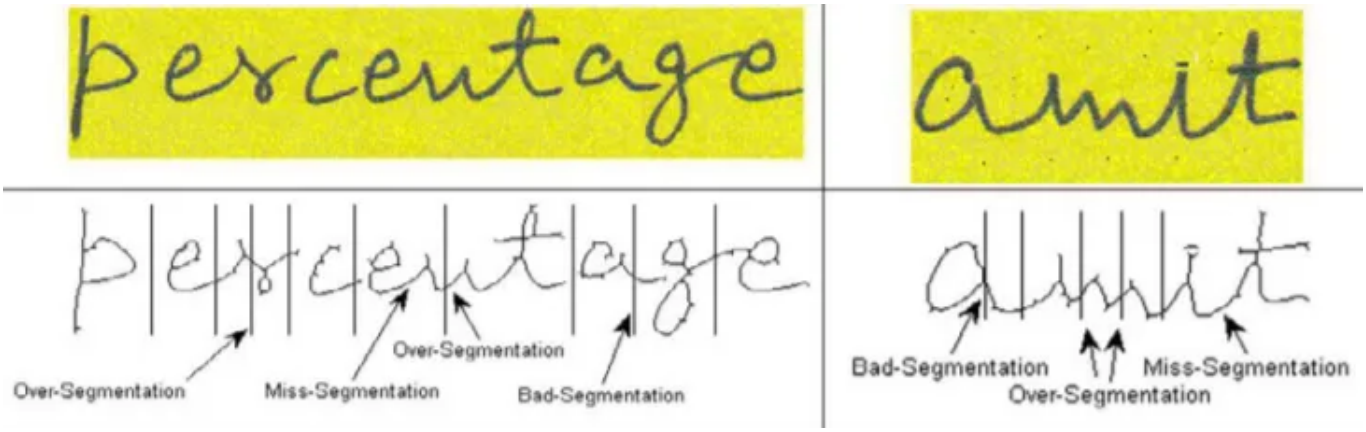
Source: Reference[1]

In the above image, Columns marked in red have either vertical histogram ***sum=0*** or ***sum=1***. These columns are known as PSC (*Potential Segmentation Columns*).

Following rules are followed to select the *Valid Segment lines* (doesn't always select a valid segment line, but a majority of times selects a valid one):-

- Clear vertical space between two consecutive characters, i.e., ***sum=0,*** along that column, indicates no foreground pixel is present. So we can select the center column among such consecutive columns.
- When there is a column with ***sum=1,*** which indicates the presence of ligature or part of open character, in this case, ***Over Segmentation problem*** is eliminated to a great extent by taking the average of all those PSC which are at a distance less than a particular value (threshold) and by merging them into a single SC(Segmentation Column). The threshold value is the minimum distance along the width of the word image between consecutive PSCs and is so chosen that its value must be less than the width of the thinnest character possible(e.g., i,l) in a word image. Now all these SC's are validated by a machine learning model trained with a huge amount of data. If the probability of an SC being a valid segmenting line is high, it is considered a valid segmenting line; otherwise, it is ignored.

By following the above rules, we can segment characters correctly 90% of the time.



Word Images Showing all types of Segmentation Errors. **Source: Reference[1]**

Final Words

In this 3-part series, we got a high-level overview of the working of an OCR System and covered the two most important phases of the OCR system *Preprocessing* and *Segmentation*.

I hope you understood and enjoyed reading this !!!

Any doubts, Suggestions, Corrections are Welcome.

Happy Learning ……Signing Off 😀 😀

References:

[1] Choudhary, Amit & Rishi, Rahul & Savita, Ahlawat. (2013). A New Character Segmentation Approach for Off-Line Cursive Handwritten Words. Procedia Computer Science. 17. 88–95. 10.1016/j.procs.2013.05.013.

[2] Sanchez, A. & Suarez, P.D. & Mello, Carlos & Oliveira, Adriano & Alves, V.M.O.. (2008). Text Line Segmentation in Images of Handwritten Historical Documents. 1–6. 10.1109/IPTA.2008.4743758.

[3] Kurniawan, Fajri & Shafry, Mohd & Rahim, Mohd & Matus, Ni & Rakhmadi, Akmal & Mohamad, Dzulkifli. (2011). Characters Segmentation of Cursive Handwritten Words based on Contour Analysis and Neural Network Validation. ITB Journal of Information and Communication Technology. 5. 10.5614/itbj.ict.2011.5.1.1.

Image Source: *Google*

Programming | Machine Learning | Ocr | Segmentation | Image Processing

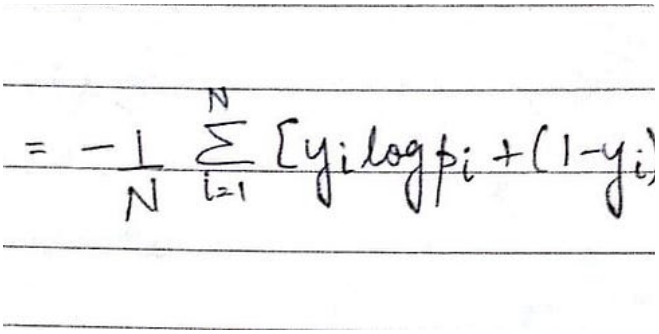
Written by Susmith Reddy

Follow

205 Followers · Writer for Towards Data Science

A lazy & reluctant writer

More from Susmith Reddy and Towards Data Science



Susmith Reddy in Analytics Vidhya

Understanding the log loss function

Gaining an in-depth understanding and intuition for log loss function from a beginne...

7 min read · Jul 6, 2020

476 7

Cristian Leo in Towards Data Science

The Math behind Adam Optimizer

Why is Adam the most popular optimizer in Deep Learning? Let's understand it by diving...

16 min read · Jan 30, 2024

1.8K 13

Siavash Yasini in Towards Data Science

Python's Most Powerful Decorator

And 5 ways to use it in data science and machine learning

· 11 min read · Feb 2, 2024

2.1K 15

Susmith Reddy in Motivate the Mind

My GRE Preparation Strategy for 326

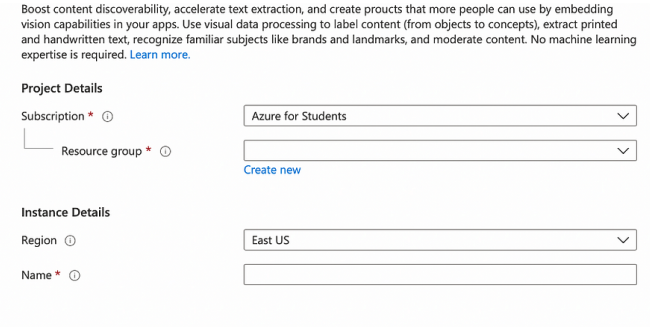
I am a full-time software engineer in India and have scored 326 in GRE. I want to share my...

10 min read · Nov 1, 2021

158 1

See all from Susmith Reddy | See all from Towards Data Science

Recommended from Medium



Buse Köseoğlu

Converting Text in Images to Text with Azure OCR

In this article, I will tell you how to convert printed or handwritten text in the image into…

5 min read · Nov 1, 2023

Mehmet Çağrı Çalpur

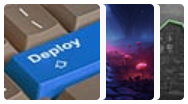
OCR with Vision Transformers

Vision transformers are disrupting the conventional visual tasks such as…

3 min read · Jan 17, 2024

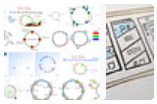
95

Lists



Predictive Modeling w/ Python

20 stories · 918 saves



General Coding Knowledge

20 stories · 929 saves

Practical Guides to Machine Learning

10 stories · 1081 saves

Natural Language Processing

1205 stories · 686 saves

Aditya Mahajan

EasyOCR: A Comprehensive Guide

Detailed explanation of EasyOCR with usage examples

11 min read · Oct 28, 2023

156

James Presbitero Jr. in Practice in Public

These Words Make it Obvious That Your Text is Written By AI

These 7 words are painfully obvious. They make me cringe. They will make your reader…

5 min read · Jan 1, 2024

38K 1023

Riwaj Neupane

OCR with PyTesseract and EasyOCR

Tesseract is an open source text recognition (OCR) Engine, available under the Apache 2....

Vinod Baste

Unlocking the Power of PaddleOCR

An Introduction to Text Detection and Recognition

4 min read · Jan 15, 2024

2

8 min read · Sep 20, 2023

85

See more recommendations