*International Journal of Computer Applications (0975 – 8887)*
*Volume 117 – No. 7, May 2015*

# Machine Learning based Multilingual OCR

Chandrahas Gaikwad
Department of Computer Engineering, SITS, Pune

Satish Akolkar
Department of Computer Engineering, SITS, Pune

Reshma Khodade
Department of Computer Engineering, SITS, Pune

Deepali Dalal
Department of Computer Engineering, SITS, Pune

Smita S. Pawar
Department of Computer Engineering, SITS, Pune

## ABSTRACT
Paperless business has led to high speed amelioration in the world of technology. Storage, processing and retrieval of data have thus become effortless. To avoid unnecessary alterations during these phases, dossiers are stored as images or as Printable Document Format (PDF). But when real time modifications are to be made, barriers occur due to platform and script dependency, leading to complications. In this project, a generic way to overcome this problem has been presented through the concept of machine learning. A learning character set and a PDF of the identical script constitute the input. The unique features of various characters in the character set are learnt by the machine through various classifiers, and a map for the same is searched in the PDF and correspondingly profiles are generated. These classifiers distinguish the characters based on number of ripples in their patterns, number of regions and other parameters. Comparison is made between both and exact match is declared as result. This project eradicates the need to 'start from scratch' for processing newly encountered script, as observed in the conventional software due to its 'classifier reuse' strategy. It touches the social aspect in situations, where data is available with the user, but in a format in which manipulation is tiresome. In such cases, user can simply give the respective PDF and its character set as input, and obtain corresponding editable version as an output.

## General Terms
Editing, Learning, Reuse, Generation.

## Keywords
Multilingual, Optical Character Recognition, Machine Learning, Classifiers.

## 1. INTRODUCTION
The world is continuously progressing taking long paces. Maintaining record of each and every move has become need of time for understanding the advancements. Also what further steps should be taken can be derived from the existing results. Earlier editable document format was used for storage of records. However with passage of time, operating environments found radical changes as computing knowledge spread like a wild fire. Owing to above circumstances, and to avoid future complexities, dossiers began to be stored in a format acceptable to all. This format was nothing but Printable Document Format (PDF). Data was also stored in the form of images. Now consider a situation where a user created a document in a specific script and later sent it to a remotely located user for evaluation. However, due to factors like compatibility, platform dependency the remotely located user could not perform the corresponding modifications. To overcome this barrier, solution can be incorporated in the form of a system which is capable of modifying the provided Printable Document Format and which is independent of the script given as input.

In this research paper, a complying solution is presented for the aforementioned problem. From the input PDF, the scripted alphabets are separated through row-wise and column wise slicing. This is followed by application of various classifiers for deciphering the separated characters and finding out its unique features. A dictionary is also used along with the classifiers in cases where ambiguities are present. E.g. The alphabets O and D have similar features. Finally, the editable text corresponding to the editable PDF is generated as output.

## 2. METHODOLOGIES

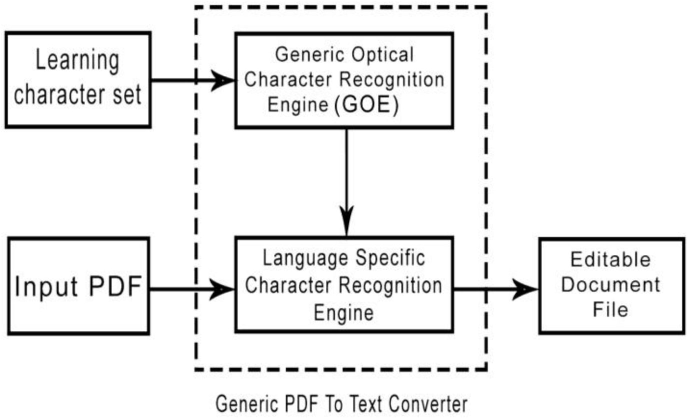


**Fig1: General block diagram of Machine Learning based Multilingual OCR**

[1]In this section a stepwise methodology is described as to how a PDF document, with the script of any language or an image can be converted into an editable document.

## 3. IMPLEMENTATION

### 3.1 Inputs
This system needs two inputs. One is the Input PDF which is to be converted into editable format and second is the learning character set consisting of various alphabets which provide a perception of the scripted alphabets in the input PDF.

### 3.2 Output
From the study of learning characters and the input PDF, an editable document file is generated as an output

### 3.3 Central Processing
The central processor is nothing but the Generic PDF to text converter itself. The two main blocks enclosed by it are: Generic Optical Character Recognition Engine and Language Specific Character Recognition Engine.

#### *3.3.1 Generic Optical Character Recognition Engine*
This block mainly administers analysis of alphabet in the input PDF through processing the learning character set.

#### 3.3.1.1 Furnishing inputs to the system

[5]Initially the system takes a learning character set as an input. This set consists of all symbols employed in the PDF provided as an input. E.g. In case of English language, the alphabets a-z should be provided as an input. Every symbol in the learning set must be in its best of state and capable of immediate processing.

#### 3.3.1.2 Processing mechanism for the learning character set

The input learning character set is acted upon by various classifier functions. A classifier function contains a set of traits which can help in distinguishing characters from one another. After processing a character, it returns values demonstrating the factor to which the character abides by the properties. The different classifiers are:

*1. Number of Regions*

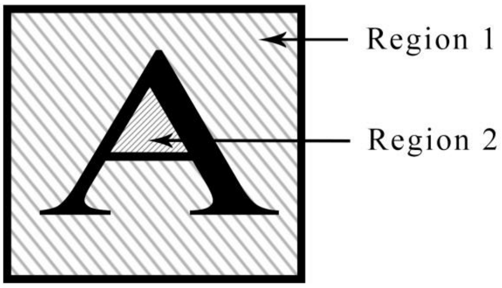


**Fig 2: Number of Regions**

The term [6]contours stands for the various points which are at the same depth or at the same level. In the context of our project, we use this classifier to calculate the number of regions present in the being processed alphabet. The steps followed are:

**Algorithm 1: Calculation of Number of regions**

1. Deciphering of all the points at the same level or depth(contours).
2. Finding out the hierarchy between the deciphered contours.
3. Calculating number of the elements in the contours' array.
4. Decrement the count by 1

The above algorithm returns contours and the hierarchy of the same for a given alphabet. Hierarchy describes the parent and child relationship between the contours at different levels. Firstly, all the contours are discerned. For understanding contour locations in the image and their relationship with each other, hierarchy is calculated. Co-ordinates of the contours are obtained in an array format, where each array element represents one contour. Now calculate the length of this array to obtain number of regions. It should be noted that the image boundaries are also termed as contours while processing. So, finally the count is decremented by 1, so as to obtain exact number of regions.

Now consider the example of alphabet 'O'. In this case three levels will be found. First will be the co-ordinates of the image. Second will be outer boundary of the alphabet and third will be inner boundary of the alphabet. In short, an additional contour is obtained. So we decrement final count by 1 and hence number of regions will be 2. In this way for every alphabet we can obtain number of regions using contours.

*2. Ripples*

Ripples describe the rise and fall in the values of any quantity. This concept can be used for processing of alphabet as follows.

Four types of ripples have been used in the processing as a part of machine learning.

a. Right ripple
b. Top Ripple
c. Left Ripple
d. Bottom Ripple

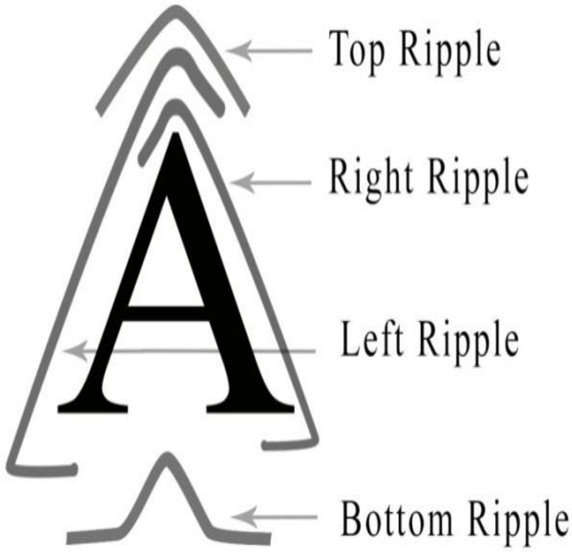


**Fig 3: Scanning of Ripples in an alphabet by the Ripple classifers**

Fig.2. explains how the ripples are generated around the character. The name associated with the ripple explains the side on which the ripple is being applied. E.g. Top Ripple is applied to the top side of the image.

The working of this classifier has been elaborated through following algorithm.

**Algorithm 2: Calculation of number of Ripples**

1. Store the ripple count in a variable, initialize it with zero.
2. Start traversing form the opposite side. E.g. if it is top ripple, start traversing pixels from the bottom side towards the top direction
3. Progress in this direction till no more dark pixels can be found, Increment ripple counter by 1 at this point.
4. Trace the contour from this junction. If the pixel co-ordinates progress at the same level, then leave the ripple count untouched.
5. If the pixel's co-ordinates considerably decrease, progress in this direction till the last dark pixel is encountered. Increment the ripple counter by 1 at this point.

To begin with, Firstly, we need a location where we can store the number of ripples encountered during the processing. Hence, the ripple counter comes into picture. Initialize the ripple counter to 0 before initiating processing. Now, begin from that side that is opposite to the name associated with the ripple. For instance, if top ripple is to be computed, then start from bottom side, facing top side. Start traversing top side till the last dark pixel is encountered. At this co-ordinate, the riser has ended, i.e. no more increment in the pixel value will be perceived here after. Hence, increment the ripple count by 1. From this point, either pixels will remain at the same depth or the depth will decrease. If the depth remains the same, the ripple count is kept unaltered. However, if the values decrease, then go on traversing in that respective direction till the last pixel is reached. Then increment the ripple counter by 1. Follow this procedure till the side on which the processing was initiated is reached.

The above procedure can be reused for computations of other ripples. E.g. Right ripple can be calculated from top ripple. Now consider that, the input image is transposed. If top ripple is calculated again for the transposed image, right ripple will be obtained. Similarly, other ripples can be calculated following this procedure.

## 3. Heaviness

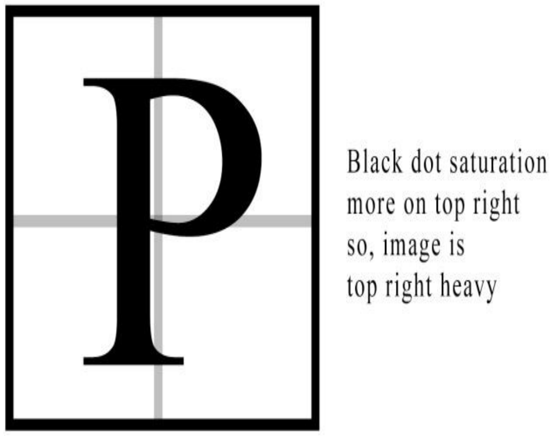


**Fig 4: Classifier for Heaviness Computation**

Heaviness describes which part of the image has more saturation of black dots. The working of this classifier undergoes following series of steps:

**Algorithm 3: Heaviness Computation**

1. Divide the input image into four quadrants.
2. Maintain a sequence for traversing the quadrants.
3. Store the number of dark pixels associated with each quadrant in an array.
4. Compare the values stored in the array. Image is heavy in that quadrant where most dark pixels are found.

An image can be divided into four parts. i.e. Left-bottom, Right-Bottom, Top-right and top-left. This procedure can be done easily in image processing with the number of rows and columns known. For traversal, a fixed sequence is denoted. Maintain this sequence throughout the processing part for each and every alphabet. When this information is completely available, simply compare the values of the four quadrants to find which quadrant has dark pixels in bulk. The one containing maximum dark pixels is the heaviest quadrant. This in turn results in the image being 'that quadrant' heavy.

For example, if heaviness of p is calculated then p has more black dots on the top right side. Thus, in this case P is top-right heavy.

### 3.3.1.3 Tree Structured working of Generic Optical Character Recognition Engine

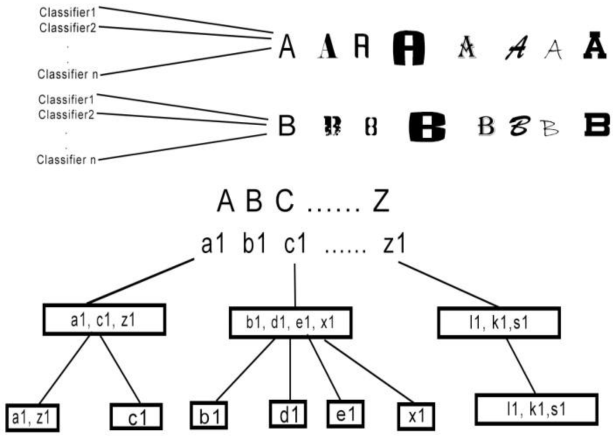


**Fig 5: Tree Structured working of Generic Optical Character Recognition Engine**

Above figure shows [7]Tree structured working of Generic Optical Character Recognition Engine. Input PDF may be in any font i.e. style of character in every font is different .Due to unique patterns in every character, values will be different for each character. These values are obtained by applying number of classifiers on each character. Values of all characters are stored in such a way that, range is defined for particular block based on values of classifiers irrespective of the values of characters themselves. The characters get separated based on their profile values into different blocks. If exactly one character is left in a particular block, then there is no need to further process it. Else, aforementioned procedure will be repeated till individual characters are obtained in individual blocks.

### 3.3.1.4 Profile Generation

[3]The values obtained from classifier functions are congregated to constitute a profile. Such profiles are created for each and every character in the learning character set as well as the input PDF. They are then converted into lists for expediting processing.

### 3.3.2 Language Specific Character Recognition Engine

As mentioned earlier, this block is generated from Generic Optical Character Recognition Engine. This step emphasizes on the second input i.e. the input PDF. It deals with slicing out the characters from the input PDF and comparison with learning character set.

### 3.3.2.1 Conversion of input PDF into binary format.

[2]The input PDF may consist of different colors. Processing of such an input will tranquilize execution considerably. Instead, the PDF can be converted to binary format so that processing on basis of two values can be initiated. This phase is carried out in two steps. Firstly, the input image is converted to grey and then the image is converted to binary. [6]The threshold value concept is used to accelerate processing and make a perfect distinction between the characters having same general characteristics.

### 3.3.2.2 Horizontal and Vertical Slicing.

[4]Next step to be taken is slice out the characters from the input. This can be obtained through horizontal and vertical

slicing. Firstly, individual rows containing text are separated from the input and then individual characters are separated from these rows. Ultimately, the outcome of this step is a character enclosed in a rectangular frame.

### *3.3.2.3 Decision Making Process*

Only those classifiers which give distinct values after application on learning character set are passed on from the Generic Optical Character Recognition Engine to Language Specific Character Recognition Engine. E.g. In fonts where all the characters are curly, the ripple classifiers may not yield distinct values. In such circumstances other classifiers like number of regions, top heavy, bottom heavy, etc. can be applied.

### *3.3.2.4 Application of Dictionary for ambiguity removal*

A dictionary is generated for the most commonly used words of the being processed language. Now consider that the word undergoing processing in English language PDF is ‘this’. Assume that there is ambiguity at third position between the alphabets ‘i’ and ‘j’. Correspondingly, two words are read, i.e. ’this’ and ‘thjs’. On referring the dictionary, the meaningful word thus present will be ‘this’. And so the resolved word ‘this’ will be displayed as an output.

### *3.3.2.5 Output Generation*

Based on aforementioned procedure, comparison between the values of learning character set and the input PDF will be done. Exactly matching results will be displayed as output. In case of ambiguity, results obtained from the dictionary are fetched and correspondingly used as output.

## 4. EXPERIMENTAL RESULTS

Working of this system and the various results obtained are elaborated as follows:

For execution an input image along with a learning character set is needed.

Inputs:

i) Input image or PDF

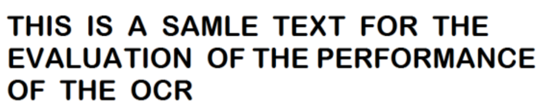


**Fig 6: Input Image**

ii)Learning Character set

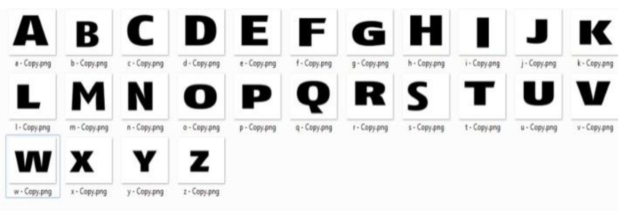


**Fig 7: Learning Character Set**

Above inputs are provided to the system for processing of English language. Various classifiers as mentioned in earlier sections of this research paper are applied on the inputs. Correspondingly profiles are generated and editable document is generated as output.

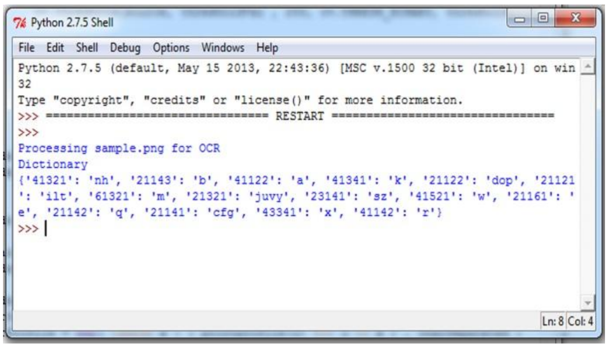


**Fig 8: Profile Generation**

Above figure denotes the generated dictionary. This dictionary constitutes profile generated for each individual character. E.g. ‘21142’ is profile for alphabet ‘q’. In some cases ambiguities are found. E.g. The profile values,’41321’ are identical for alphabet ‘n’ and ‘h’. In such situations, the system will refer the dictionary of that specific language, to resolve these ambiguities.

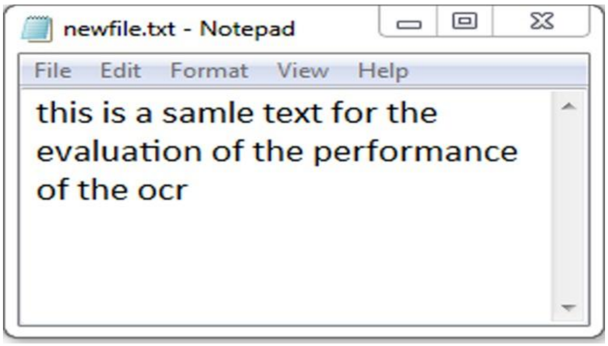


**Fig 9: Output in Editable format**

Above fig. denotes the output text file in editable format. This text is ready for all types of manipulation works which are easily performed on conventional text

## 5. FUTURE SCOPE

Data stored in images has its scope limited to fonts of the technological world. However, handwritten fonts have no limitations, as any alphabet can be written in number of ways. In such situations, it is difficult to classify alphabets. Characters may not be unique and number of regions may vary as number of joints in the alphabet may differ. The research work related to this phenomenon will be presented in the future works of this system. Quality and light intensity are some other factors to be taken under consideration. The quality of the inputs has a considerable effect on overall performance of the system. If the quality of input is not processing-sufficient, then supplementary dots or noise may degrade the system’s performance considerably. A proper discriminating algorithm for textual characters and noise is planned to be devised in the future research works of this system.

## 6. CONCLUSION

In accordance to the various implementation techniques and application of various test cases on them, Following conclusions were drawn.

Machine Learning feature overcomes the problem of redesigning similar solution set for every script unknown to the system. A classifier function performs the role of character discrimination based on their unique patterns. This research paper emphasizes on 'classifier reuse' strategy i.e. Processing of different scripts will be done by same set of classifiers. Outcome of the aforementioned procedure is the Language Specific Character Recognition Engine which consists of classifiers specific to that particular script. Application of dictionary resolves the problem created by the ambiguity between alphabets of similar nature. This helps in boosting the accuracy of the system.

**Table 1: Comparison between traditional and existing PDF to Text Converters**

| System | Generic | Working mode | Classifier Reuse Strategy | Efficiency |
|---|---|---|---|---|
| Online PDF to text Converters | ✗ | Online | ✗ | 100% |
| Offline PDF to text Converters | ✗ | Offline | ✗ | 100% |
| Machine Learning based Multilingual OCR | ✓ | Online/ Offline | ✓ | 99.9% |

The above chart demonstrates comparison between conventional software Machine Learning based multilingual OCR. Conventional Online and offline software are not generic, i.e. if an unknown script is given as input, then processing would be cumbersome. Efficiency provided is maximum in most of the cases. In case of Machine Learning based Multilingual OCR, working mode is both online and offline Efficiency is maximum, however it may be hampered as machine learning is a complex process as compared to the straight-forward approach. Output is generated as a .doc file, which can be edited through any text editor.

## 7. REFERENCES

[1] Text Classification Using Machine Learning Techniques, M. IKONOMAKIS, S. KOTSIANTIS, V. TAMPAKAS

[2] Machine Learning for Image Classification and Clustering Using a Universal Distance Measure, Uzi Chester and Joel Ratsaby, Electrical and Electronics Engineering Department, Ariel University of Samaria, ARIEL 40700

[3] Cursive character recognition – a character segmentation method using projection profile-based technique Roberto J. Rodrigues, Antonio Carlos Gay Thomé

[4] A Two Stage Classification Approach to Tamil Handwriting Recognition. S. Hewavitharana, Department of Computer Science, University of Colombo, Colombo 03, Sri Lanka, H. C. Fernando, Sri Lanka Institute of Information Technology, Colombo 03, Sri Lanka

[5] Peter W. Frey and David J. Slate,"Letter Recognition Using Holland style Adaptive Classifiers" Department of Psychology, Northwestern University, Evanston, IL 60208

[6] A Simple and Effective Optical Character Recognition System for Digits Recognition using the Pixel-Contour Features and Mathematical Parameters, Jenil Shah, Viral Gokani.

[7] Tree Structured Data Analysis: AID, CHAID and CART Leland Wilkinson, SPSS Inc., 233 South Wacker, Chicago, IL 60606, Department of Statistics, Northwestern University, Evanston, IL 60201