Optical Character Recognition System for Urdu (Naskh Font) Using Pattern Matching Technique

Tabassam Nawaz drtnawaz@uettaxila.edu.pk
Faculty of Software Engineering,
University of Engineering & Technology
Taxila, Pakistan

Syed Ammar Hassan Shah Naqvi ammar_hassan13@yahoo.com
Department of Software Engineering
University of Engineering & Technology
Taxila, Pakistan

Habib ur Rehman habieb_rehman@yahoo.com
Department of Software Engineering
University of Engineering & Technology
Taxila, Pakistan

Anoshia Faiz an_oshia@yahoo.com
Department of Software Engineering
University of Engineering & Technology
Taxila, Pakistan

Abstract

The offline optical character recognition (OCR) for different languages has been developed over the recent years. Since 1965, the US postal service has been using this system for automating their services. The range of the applications under this area is increasing day by day, due to its utility in almost major areas of government as well as private sector. This technique has been very useful in making paper free environment in many major organizations as far as the backup of their previous file record is concerned. Our this system has been proposed for the Offline Character Recognition for Isolated Characters of Urdu language, as Urdu language forms words by combining Isolated Characters. Urdu is a cursive language, having connected characters making words. The major area of utility for Urdu OCR will be digitizing of a lot of literature related material already stocked in libraries. Urdu language is famous and spoken in more than 3 big countries including Pakistan, India and Bangladesh. A lot of work has been done in Urdu poetry and literature up to the recent century. Creation of OCR for Urdu language will make an important role in converting all those work from physical libraries to electronic libraries. Most of the stuff already placed on internet is in the form of images having text, which took a lot of space to transfer and even read online. So the need of an Urdu OCR is a must. The system is of training system type. It consists of the image preprocessing, line and character segmentation, creation of xml file for training purpose. While Recognition system includes taking xml file, the image to be recognized, segment it and creation of chain codes for character images and matching with already stored in xml file.

The system has been implemented and it has 89% recognition accuracy with a 15 char/sec recognition rate.

Keywords: Pattern matching, chain code creation, morphology, segmentation, training system, recognition system, digital image processing.

1. INTRODUCTION

An Optical Character Recognition System is software engineered to convert hand-written or typewritten text (usually scanned) documents into machine editable text formats.

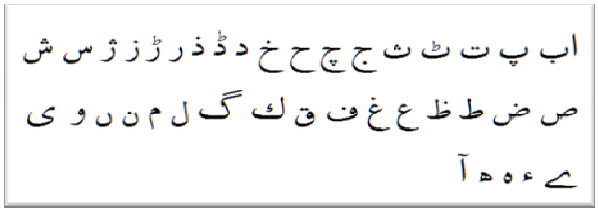

FIGURE 1: Urdu Characters set

This paper describes a training based offline [11] optical character recognition system for a Naskh font of Urdu language. The main idea behind this recognition is matching the pixel values of the samples already stored with pixel values of those character-images to be recognized. The major difficulty while the recognition of Urdu language is its cursive nature, i.e. the characters joined to create new words. The research in this has been greatly increased during last decade, as the applications of this area are increasing. It has improved Human Computer Interaction; other examples include paperless environment, online newspapers, old literature online availability, paper checking, automating official tasks, reading bank receipts, postal addresses and data entry forms. Many people are now a day working on Urdu OCR research. As Standard Urdu has approximately the twentieth largest population of native speakers, among all languages. Due to technical issues induced by the cursive nature of Urdu language, its OCR has not been developed completely. If Urdu OCR system is available, it will be very useful and will have great commercial value. The overall flow of our OCR is shown in Fig.2. Section II describes the Urdu language specifications. Some related work done for Urdu OCR is described very briefly in Section III. Section IV and V describes the Training and Recognition Systems respectively. Section VI describes the process of creating Unicode file [3] from already recognized characters. All the algorithms used in this paper are described in Section VII. Finally, this paper is concluded in Section VIII.

2. URDU LANGUAGE SPECIFICATIONS

Urdu language is the old language of Indo-Pak Sub-continent, now it is the national language of Pakistan. This language is a combination of characteristics of Arabic, Farsi and Sanskrit languages, as it is the language of troops. The Urdu language has the characteristics of all these languages mentioned above, all the characters in this language are picked from these languages. Urdu language is a more cursive and complex language than Arabic and Farsi language as it contains the connected characters to make words. This cursive nature [9] makes it very difficult to be recognized through usual Character Recognition Methods.

Urdu character set consists of 40 characters. The characters contain single loop, double loops and incomplete loops. Dots and diacritics (i.e., Telda (~) and Nuktay (◌)) are also included in the character set. Dots include single, double and triple dots. The recognition of Urdu language is

very difficult due to the different multiple shapes of a single character. In Urdu language, every letter is of minimum 2 shapes and maximum 4 shapes. These shapes are based on their occurrence in the given word. The locations in the Urdu word are isolated, initial, medium and last. Here all isolated shapes of the characters are mentioned in Fig.1. So the Urdu language character recognition is very difficult still. Another characteristic of this language is; if we read or write something is from right to left.

3. RELATED WORK

One of the oldest techniques for pattern recognition is used for character recognition, but through all days, more focus was on Latin, Chinese [24] and Japanese [25] languages, though connected. First, we applied Hilditch's method, which consists of removing the pixels that lie on the edge of the binary image until only one-pixel-wide line remains. This is followed by some conditions suggested by Al-Emami to reduce the junction points to one junction point [23]. The matching procedure is executed based on an image based matching algorithm. From a practical viewpoint however, the matching time must be reduced as much as possible through the classification techniques [3].

In our studies we analyzed the shape and visual properties of Urdu characters and define a set of features which can distinguish one character to another. In Urdu Qaeda system, the online character recognition technique is used for isolated characters which is some how based on the strokes by the user runtime [17].

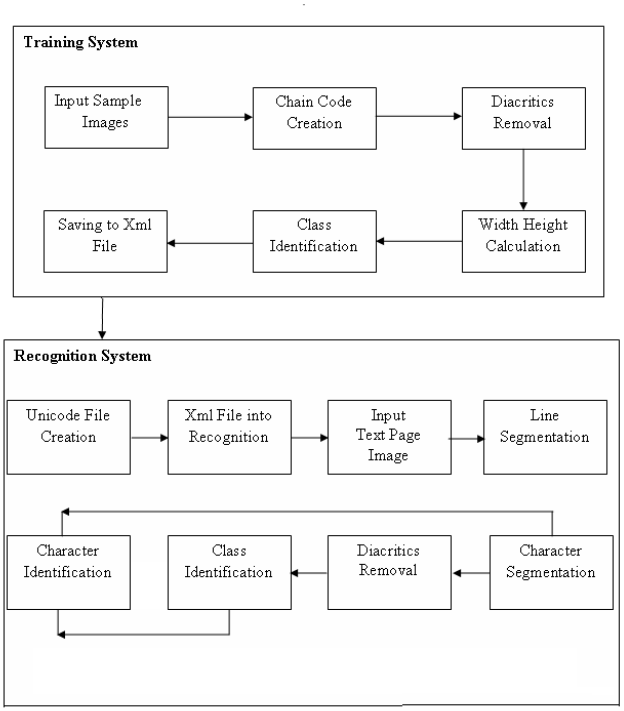

FIGURE 2: Block Diagram of System

4. SYSTEM

4.1 Training System

- *Requirements*

As we will first train our system for a specific font and then matching algorithm will match both the sampled image information and the actual image (input image) information. So the system needs to know the font and size of the text to be recognized. Thus we have to select a specific font and specific size in our system. In our case, the following demonstration of training and matching is only for NASKH font and it can be extended to any font of any size. Sample font size to demonstrate is taken to be 36. The images used for training should be noise-free and gray scale. We will first convert the gray scaled image into binary image as shown in Fig.3, and remove extra noise from the image. Now we can apply morphological transforms [12] easily on these images to get required results.


Original Image


Binary Image

FIGURE 3: Binarization of input image

- *How to Take Character's Sample*

A gray scaled image comprising of the whole character set of Urdu language (Naskh font) is taken as sample and individual characters are separated out. All these individually separated characters are created in such a way that from each side one totally white column or row is left for the ease of chain code [2] creation. As stated earlier, only isolated characters will be recognized so the samples are taken in such a way. This technique can be applied to any font having any size.

- *How to Create Chain code of characters*

The individual character images are then passed into the training software. Each character image is then scanned from top to bottom and then to the next column, a chain code [18] of each column of the image is generated. The chain code based on the sum of consecutive one's or zeros, as the image is only a binary image. According to our assumption, Zero is assumed to be the first entry of the column. If this Zero appears zero times, then we will place 00 in the string, then for example 1's appear for 23 times, we will add 23 in the string. If we continue scanning, now its turn of zero, if zero appears for 12 times, we will add 12 to the string. Now, if the column is finished, we will add @ to the string to make sure that column is ended (as shown in Fig.4) the same process will be repeated for next column and so on. It should be noticed that when the number is less than 10 e.g., 3 zeros then we write "03"; we add this because maximum no. of pixels can't exceed 99. As these columns are of only individual characters. Thus by calculating all columns we get a chain code (from the final string) of set of the character image. This chain code is generated by calculating alternating on and off pixels as shown in the Fig.4.

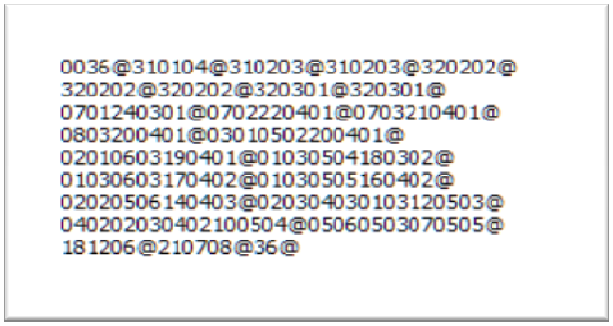

FIGURE 4: Calculated String

Now the chain code is available, the task is to store this chain code, we can use different options for storing this string including database, collection objects of C# and xml file. We have chosen the xml file for storing this file, as it will take least space and reading/writing into the file will be easy. The Fig.5 explains the syntax of our xml file. This xml file is described in next section.

FIGURE 5: Classxml.xml

- *How to create classes*
- *Xml file*

Second step during training phase is creation of xml file. Xml file contains all the 21 classes [3] of Urdu alphabets as parent nodes or elements. Each class contains character set belonging to that particular class, where every character makes a single child node of parent class node. Each child node has three attributes. One, the name of the character and the other, chain code of that character, calculated from its image earlier. Unicode of the character is saved in xml as third attribute of the child node, which will be assigned to the identified character at the end of the matching procedure.

- *Classification Criteria*

Classification of characters is the key technique in our pattern matching method of optical character recognition. Urdu script has a large character set, consisting of 40 characters. Pattern matching technique is of no use if system has to traverse all the 40 characters and match their chain codes. This special matching technique [14] is made very efficient by optimization at every level, including logical methods and programming techniques. This classification is done on the bases of height and width of the character without diacritics and dots. Urdu script consists of total 21 shapes, occupying specific dimensions. These 21 shapes are declared as the classes in which our system divides the characters for recognition. All the recognize-able characters are input to the training module to train our OCR system for a specific font and size. Training system can train

the OCR for any font and size character set, as mentioned earlier. System will store their dimensions as individual classes of characters. Fig.6 describes all the 21 classes.

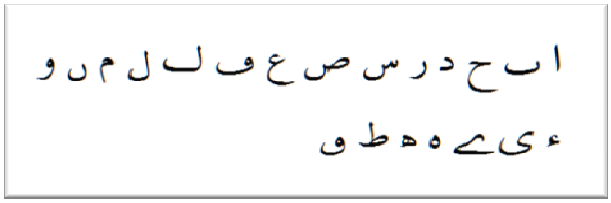

FIGURE 6: Character's Classification

- *How to remove diacritics*

The character's dots and diacritics are removed using morphology techniques [12] of binary images. The dots are removed using pepper noise [6] removal process and diacritics like ~ are removed using thinning process [5]. Special filters are designed for removing diacritics. In Urdu script characters contain a variation of diacritics, like single Nukta, double Nukta or triple Nukta and "Chota Tuay" as in Fig.7.

FIGURE 7: Shows Chota Tuay, Double, triple and Double Dots respectively from left.

- *Filters*

Different special filters are designed particularly to remove diacritics from the isolated character images. When the shape of the character is obtained after removing diacritics, we calculated the dimensions (height and width) of each character and place it into particular class. Fig.8, Fig.9a, Fig.9b and Fig.9c describe the filter for removing ttuay' and single dot, double dot, triple dot respectively.

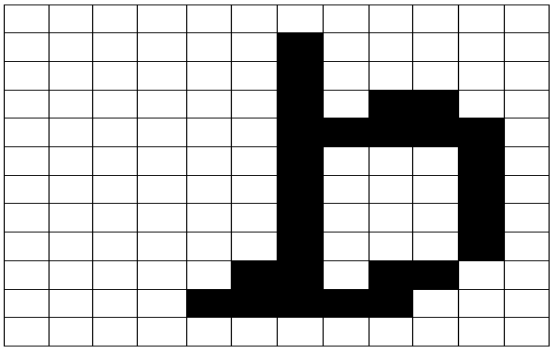

FIGURE 8: Ttuay Filter

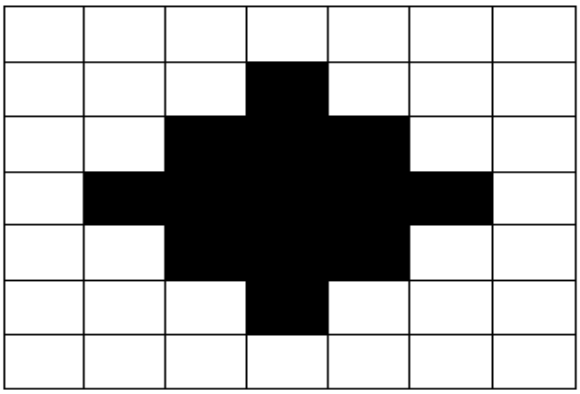

FIGURE 9:. Single dot Filter

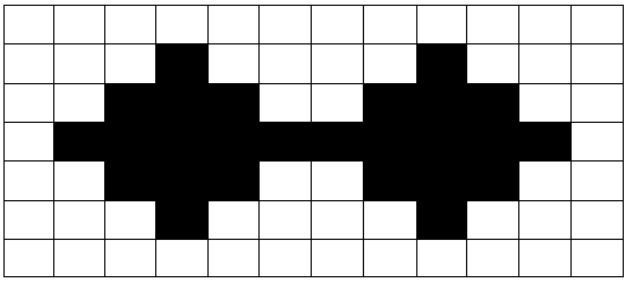

FIGURE 9a: Double dots Filter

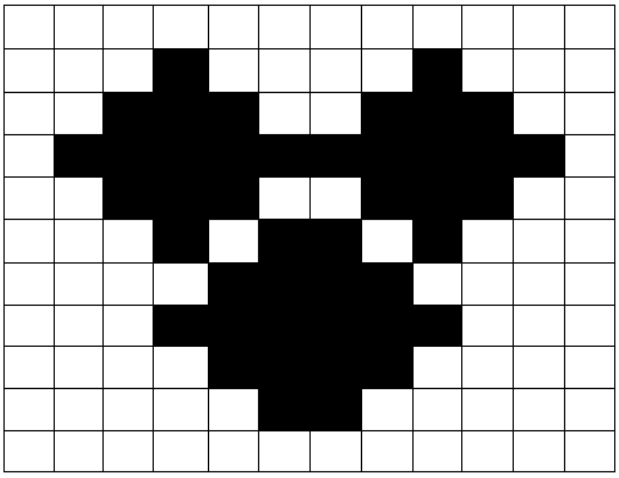

FIGURE 9b: Triple dot Filter

- *Chain code*

After creating the xml for 21 classes, measured the chain code for each class characters. All the characters of a class are added to the xml file as children to the parent classes. All the character set classes are shown in the Fig.1. This xml file will act as an input for the recognition system.

4.2 Recognition System

- *Segmentation*

When an image containing text is given as the input to the character recognition system, system performs some preprocessing steps on it; including converting grayscale image into binary image and enhancing the image by removing pepper noise. Binary image is need for applying filters and

calculating chain code. Pepper noise must be removed because it will lead the system to erroneous classification and character recognition. Error margin (described later in this section) methods are very sensitive to extra noise in image as noise can be located at random locations disturbing the chain code and specially dimensions. Remember that "character image variation" is always symmetric, not random. System then divides the image into segments. As at this stage this paper is describing the recognition of isolated characters, thus segmentation is required at two levels instead of three levels as required for connected characters in recursive script i.e. Urdu script.

Two levels of segmentation are:

- *Line segmentation*

The image is first divided into different lines of text by checking for whole row of white pixels consecutively from right to left. Each separated line will be saved separately in the recognition system. Fig.10 shows an image after the lines have been segmented from an image.

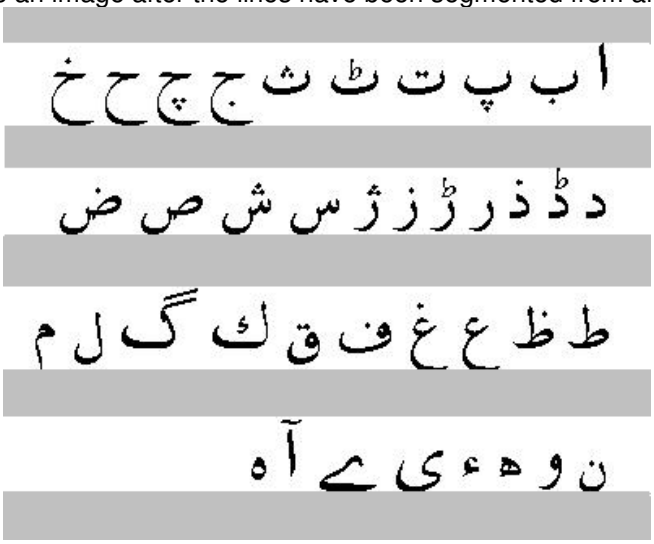

FIGURE 10: Line Segmentation Results

- *Isolated character segmentation*

The already separated text line images are to be processed now, for isolated character recognition, by converting them into individual characters. During processing, we check for full white pixels column at the starting and ending point of the isolated character. Start of the character is identified when any black pixel is scanned in the column. Scan continues until another white pixel column is identified. Image in-between the white columns, is saved, and starting and ending white column of pixels is preserved in the image. These results into different small images of individual characters are shown in Fig.11 which will be processed separately. Every separate image of isolated character is now completely void on four sides. Leaving one pixel margin on each side of the image is because of the assumption that in chain code of each column of the image will start with white pixels count.

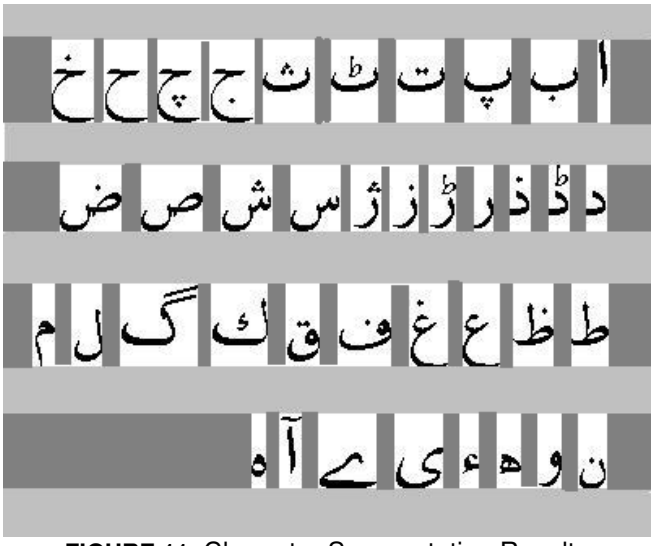

FIGURE 11: Character Segmentation Results

- *How to Identify Class of a Character*

After segmentation of whole document image into small images of isolated characters, next step is to identify class of each character. Following Steps will describe whole recognition steps;

- *Diacritics Removal*

To obtain an image of recognizable class, dots (Nuka's) and diacritics (ttuay) are removed from the above, below or inside the character image using especially designed filters (as shown earlier in Figs[8,9a,9b,9c]) specific to them. This dot-less image is stored as another copy of the character image. Now, the height and width of this new character image are measured with only considering black pixel, as was done before while taking samples and matched with the height and width of different classes in the classxml.xml file.
When a class gets matched, then the image's string will be calculated again and matched with the already calculated strings of the matched class characters each. This classification helps in providing more efficient approach in pattern matching.

- *Chain code calculation*

At this stage, we knew the class to which the character belongs; now the original image will be used for recognition. And this original image is scanned from top to bottom for each column to obtain its' chain code. We will call this chain code, the "calculated chain code" and the one that is saved in characterset.xml file called "sampled chain code". The calculated chain code is calculated using the scanning the input image and generating the calculated string of on and off pixels as described earlier in section 3.1.

- *How to Recognize Character*

After determining the class of the character, we have to just match the calculated chain code with only some character's sampled chain codes. This makes the process of identification more reliable and efficient. The calculated chain code is matched column by column with every character's sampled chain code in the identified class. This process is repeated until all the columns are matched with some extent of error margin. As we have already set the error margin, character is identified up to the margin because different images can have little different properties, so exact pattern matching cannot be so efficient, to identify the exact character.

  - *How to check Error Margin*

As any image can be error prone, we must satisfy the "chain code matching method" about the correct and exact character to recognize. In any case error count (mismatches) for each column is calculated, while matching total number of columns that is the width of the image. If width varies, equalize sample image width with the calculated image width. To equalize the width just add some characters like '.' in the shorter image as new columns. Now both images contain equal number of columns to match with each other. Once we are done with width it's time to check whether both columns are of equal height or not. To equalize the heights of sample column and calculated column again add characters like '.' in shorter image as a new row in affected column. Now both images contain equal number of columns and in each column equal number of rows. Match column by column the chain code and calculate the mismatches as error count. If error count increases the already set value, get to the next image in the identified class only. Character is identified when error is less than error margin.

5. ALGORITHMS USED

Chain Code Calculation

- Start at right top of the image.

- Scan from top to bottom of the first column of segmented image.

    - Assume that first pixels scanned are "off", so count number of "off" pixels and add to chain code.
    - Continue scan, if pixel value changes from "off" to "on", start counting "on" pixels and add to chain code.
    - Continue scan, counting "on" and "off" pixels till bottom of the image is reached and add to chain code.
    - At the end of the first column code add "@" as the end of the column.
    - Continue scanning the next columns to the width of the segmented image. Chain codes of all columns are calculated and saved as a single chain code in the xml file, in the corresponding class.

Segmentation

Line Segmentation:

- Load input image.
- Start scan from right top of the original image.
- Scan up to the image width, on the same Y component.
- Scan the first row to check any "on" pixels. If no "on" pixel is found go to next row.
- Continue scan until a row containing "on" pixels come across. Save the row before this row as the top edge of the image text line.
- Continue scan until another row with all "off" pixels is found. Set this row as the bottom edge of the image text line.
- Save image as first text line and continue scan to find next line.
- Continue till the bottom of the image is reached. Save all text line images to be input to character segmentation

Character Segmentation:

- Input text line image.
- Start scan from right top of the text line image.
- Scan up to the image height, on the same X component.

- Scan the first column to check any “on” pixels. If no “on” pixel is found go to next column.
- Continue scan until a column containing “on” pixels come across. Save the column before this column as the right hand edge of the isolated character image.
- Continue scan until another column with all “off” pixels is found. Set this column as the left hand edge of the isolated character image.
- Save image as first character and continue scan to find next character.
- Continue till the left hand edge of the image text line is reached. Save all character images.

Classification

- Load the segmented character image.
- Apply filters to remove diacritics, and get the shape of the corresponding class.
- Measure the dimensions of the shape i.e. the width and height of the character image without diacritics, by scanning the image from top right to left bottom pixel by pixel.
- Open xml file and traverse all the class nodes. Match “width” and “height” attribute values with the calculated values.
- When a dimension match occurs according to some already set error margin (in this case error margin is 1, i.e. if width or height is one less or on greater than the sample values, match occurs.) set that class as the key class for that character.

Character Matching

- Restore the “original segmented character image” with diacritics and dots. Calculate its chain code according to the algorithm described earlier in this section.
- Open xml file. Traverse all child nodes in the identified class and match “code” attribute values with the calculated chain code.
- When a chain code match occurs according to Error Margin, described in next algorithm (Testing), set the “name” attribute of that node as the name of the character. And set the “Unicode” attribute of that node as the Unicode of the input character.

Testing

- Get the sampled chain codes in the identified class from xml file and the calculated chain code of the classified character image.
- Start matching the calculated chain code with each of the sampled chain codes.
- Chain code of each column is compared.
- In each column every single code is compared with actual value (sampled value). In each column every single mismatch is counted as “error count”. If error count exceeds the Error Margin, set according to font and size, disqualify the node to be the exact match.
- Continue with the next node chain code and check the Error Margin. Continue until a character is confirmed. Confirmation occurs when “total error count” of the character is less than the “Error Margin Limit”.

Unicode File Creation

- Characters are recognized through the above process.
- Xml file includes all the information about the recognized character.

- This recognized character is written into a Unicode file, according to the information written in the xml file.

- The Unicode file is written from right to left.

6. RESULTS

Unicode is simply a character encoding system. It has nothing to do with how these characters finally get displayed on computer screen. Thus it cannot be Naskh based only, its only type style issues relating to rendering which is handled by operating system and application not by encoding scheme [4]. After getting the identification of exact character, we create a new Unicode file that will include all the characters identified. All the Unicode characters are according to the Unicode Consortium [3]. We will place the Unicode of those characters in the same position from right to left as in the Urdu text image file. The output Unicode file is now editable and searchable. This file is written from right to left as Urdu language is right to left.

7. CONCLUSION

This training system has recognized different printed Urdu text image files, which include isolated characters. The training system has been tested for different sizes of fonts, the results are quite impressive. Our system has achieved an accuracy of 89% for the isolated characters with accuracy of a 15 char/sec recognition rate. Different sizes of filters are used for different sizes, and this system has proven good for multi font sized characters. The system has been tested for different images and efficient results are found. Fig12.a shows the input image file, after being processed by our OCR, the screen shot of our output text file is shown in Fig.12b.

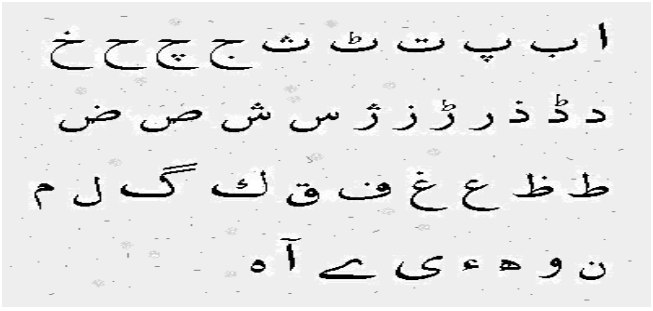

FIGURE 12a: Input image

ا پ ت ٹ ث ج چ ح خ د ڈ ذ ر ڑ ز ژ س ش ص ض ط
ظ ع غ ف ق ك گ ل م ن و ھ ء ی ے آ ہ

FIGURE 13: Output text file

8. REFERENCES

[1]. Afzal, M. and Hussain, S., "Urdu Computing Standards: Urdu Zabta Takhti (UZT) 1.01", in the Proceedings of International IEEE Multi topic Conference (INMIC), Lahore University of Management Sciences (LUMS), Lahore, Pakistan, 2001.

[2]. Ethnologue, Languages of Pakistan, http://www.ethnologue.com/show_country.asp?name=Pakistan

[3]. See the Unicode Consortium website at http://unicode.org
[4]. Bhurgari, A. M. 2007. Enabling Pakistani Languages through Unicode, published at http://download.microsoft.com/download/1/4/2/142aef9f-1a74-4a24-b1f4-782d48d41a6d/PakLang.pdf
[5]. Thresholding, Image Segmentation, Digital Image Processing 2/e Rafael C. Gonzalez, Richard E. Woods.
[6]. Fast, Bruce B., Allen, Dana R. OCR image preprocessing method for image enhancement of scanned documents.
[7]. Zaheer Ahmad, Jehanzeb Khan Orakzai, Inam Shamsher, and Awais Adnan. "Urdu Nastaleeq Optical Character Recognition", "Proceedings of world academy of science, engineering and technology volume 26 december 2007".
[8]. U. Pal and Anirban Sarkar, "Recognition of Printed Urdu Script", "Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003)".
[9]. Khalid Saeed, "New Approaches for Cursive Languages Recognition: Machine and Hand Written Script and Texts".
[10]. T. Sari and M. Sellami, "Cursive Arabic Script Segmentation and Recognition System".
[11]. Bozinovic, R.M.; Srihari, S.N, "Off-line cursive script word recognition".
[12]. Soille, P. [2003]. "Morphological Image Analysis: Principles and Applications", 2nd ed., Springer-Verlag, NY.
[13]. Dougherty. E. R. and Lotufo, R. A. [2003]. "Hands-on Morphological Image Processing", SPIE--The International Society for Optical Engineering, Bellingham, WA.
[14]. International Journal of Pattern Recognition and Artificial Intelligence.
[15]. Alasdari McAndrew, Anne Venables, "A 'Secondary' Look at Digital Image Processing".
[16]. Ganapathy, V., Lean, C.C.H., "Optical Character Recognition Program for Images of Printed Text using a Neural Network".
[17]. Nabeel Shahzad, Brandon Paulson, Tracy Hammond, "Urdu Qaeda: Recognition System for Isolated Urdu Characters".
[18]. Hermilo, Ernesto, Ramon M. "Efficiency of chain codes to represent binary objects".
[19]. Yong Kui Liua and Borut Žalik, "An efficient chain code with Huffman coding".
[20]. Shah, Z.A., "Ligature based optical character recognition of Urdu- Nastaleeq font".
[21]. Inam Shamsher, Zaheer Ahmad, Jahenzeb Khan Orakzai and Awais Adnan, "OCR For Printed Urdu Script Using Feed Forward Neural Network".
[22]. "The Origin of Urdu Language" http://www.essortment.com/all/urdulanguage_rguo.htm
[23]. T.S El-Sheikh and R.M Guindi, "computer Recognition of Arabic Cursive Script," Pattern Recognition, Vol.21, No, 4, 1988, pp.293-302.
[24]. G. Nagy Rensselaer Polytechnic Institute Troy, New York, "Chinese Character Recognition A Twenty Five Year Retrospective". Tsuyoshi Kitani t, riguchi and Masami Ilara Yoshio, "Pattern Matching in the Textract Information Extraction System".