



# **Urdu Optical Character Recognition System**

## **MS Thesis**

Submitted by

Ahmed Muaz (07-0907)

Submitted in Partial Fulfillment of  
the Requirements for the  
Degree of

**Masters of Science (Computer Science)**

Department of Computer Science  
National University of Computer & Emerging Sciences  
Lahore, Pakistan

*Approved*

---

*Head of Computer Science Department*

*Approved by Committee Members*

***Supervisor***

---

*Dr. Sarmad Hussain  
Professor, CS Department,  
National University of Computer  
and Emerging Sciences, Lahore*

***Co-Supervisor***

---

*Dr. Mehreen Saeed  
Assistant Professor, CS Department,  
National University of Computer  
and Emerging Sciences, Lahore*

*The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled "Urdu Optical Character Recognition System" by Ahmed Muaz in partial fulfillment of the requirements for the degree of Master of Science.*

*To My Parents*

## VITA

Mr. Ahmed Muaz received a Bachelor of Science degree in Computer Science from National University of Computer and Emerging Sciences (NUCES), Lahore in 2007. He has been a member of CRULP since 2005 and working as a development engineer in PAN Localization project. The research in this Dissertation was carried out from 2008 to 2010.

## Acknowledgments

First of all I would like to thank Almighty Allah who gave me strength and courage to accomplish this work.

I am grateful to my advisor Dr. Sarmad Hussain as well, for his kind considerations, encouragement, supervision and help throughout the course of this research work. I am also thankful to him for trusting me and for all the facilities he provided me to complete my research.

I am thankful to my co-advisor, Dr. Mehreen Saeed for the guidance and help. She has been a source of inspiration and kind help for me.

I am also thankful to CRULP for providing me an opportunity to develop this research work. I must thank my colleagues at CRULP as well specially Mr. Umer Khalid and Ms. Sobia Tariq for helping me in the implementation of my system.

I am thankful to my life partner Rabiah, for encouraging me to finish my research and her valuable support throughout this work acted as a catalyst.

And last but not least, I am grateful to my parents for their hidden help in terms of prayers and faith in me, which actually empowered me to fulfill this task.

**Ahmed Muaz**

# Contents

<b>CHAPTER 1.</b>	<b>INTRODUCTION.....</b>	<b>9</b>
<b>CHAPTER 2.</b>	<b>URDU WRITING SYSTEM .....</b>	<b>10</b>
2.1.	INTRODUCTION OF URDU WRITING SYSTEM .....	10
2.2.	ALPHABET .....	10
2.3.	DIACRITICS .....	11
2.4.	CLASSIFICATION OF ALPHABET .....	11
2.5.	WRITING STYLES .....	12
2.6.	DETAILED ANALYSIS OF NASTALEEQ .....	12
2.6.1.	<i>Diagonality and Stacking</i> .....	13
2.6.2.	<i>Hidden baseline</i> .....	13
2.6.3.	<i>Joining and shape conversion</i> .....	14
2.6.4.	<i>Overlapping</i> .....	15
2.6.5.	<i>Diacritic placement and shifting</i> .....	15
2.7.	NOORI NASTALEEQ .....	16
<b>CHAPTER 3.</b>	<b>LITERATURE REVIEW .....</b>	<b>17</b>
3.1.	OCR SYSTEM.....	17
3.1.1.	<i>Importance of OCR</i> .....	17
3.2.	GENERIC OCR MODULES .....	17
3.2.1.	<i>Preprocessing</i> .....	17
3.2.2.	<i>Segmentation</i> .....	19
3.2.3.	<i>Recognition system</i> .....	19
3.2.4.	<i>Post processing</i> .....	19
3.3.	HISTORICAL BACKGROUND (CURRENT STATE OF OCR SYSTEMS) .....	20
3.3.1.	<i>English OCR</i> .....	20
3.3.2.	<i>Arabic OCR</i> .....	21
3.3.3.	<i>Urdu OCR</i> .....	23
<b>CHAPTER 4.</b>	<b>PROBLEM STATEMENT .....</b>	<b>26</b>
4.1.	SCOPE .....	26
<b>CHAPTER 5.</b>	<b>PROPOSED METHODOLOGY .....</b>	<b>27</b>
5.1.	DESIGN MODEL.....	27
5.2.	MAIN BODY AND DIACRITICS ISOLATION .....	29
5.3.	THINNING (SKELETONIZATION).....	29
5.3.1.	<i>Process definition</i> .....	29
5.3.2.	<i>Process details</i> .....	29
5.4.	SEGMENTATION.....	30
5.4.1.	<i>Starting point</i> .....	30
5.4.2.	<i>Segmentation procedure</i> .....	31
5.4.3.	<i>Restoration of original segment</i> .....	32
5.4.3.1.	<i>Filling</i> .....	32
5.4.3.2.	<i>Cutting</i> .....	33
5.4.3.3.	<i>Challenges</i> .....	33
5.4.4.	<i>Segmentation algorithm</i> .....	34
5.5.	FRAMING & DCT .....	34
5.5.1.	<i>Framing need and selected methodology</i> .....	34
5.5.2.	<i>Framing process</i> .....	35

5.5.3. DCT.....	36
5.6. RECOGNITION.....	36
5.7. LIGATURE CONSTRUCTION (POST PROCESSING) .....	37
5.7.1. Segment reordering .....	37
5.7.2. Diacritic association .....	38
5.7.3. Unicode mapping .....	38
<b>CHAPTER 6. TRAINING &amp; TESTING METHODOLOGY .....</b>	<b>41</b>
6.1. TRAINING DATA .....	41
6.2. TRAINING PROCEDURE.....	41
6.2.1. Ligature extraction .....	41
6.2.2. Unique ligature selection.....	43
6.2.3. Sample preparation and segments generation.....	43
6.2.4. Clustering.....	43
6.2.4.1. K-Means.....	43
6.2.4.2. HTK based training with manual pruning .....	44
6.2.5. Model training.....	45
6.3. TESTING DATA .....	46
6.4. TESTING APPLICATION .....	46
6.5. TEST PLAN .....	47
<b>CHAPTER 7. RESULTS AND ERROR ANALYSIS.....</b>	<b>48</b>
7.1. SEGMENTATION.....	48
7.1.1. Issues.....	48
7.1.2. Suggestions.....	50
7.2. RECOGNITION.....	50
7.2.1. Issues.....	51
7.2.2. Class-wise comparative analysis.....	52
7.3. POST PROCESSING ERRORS (LIGATURE LEVEL).....	53
7.4. SUMMARY.....	56
<b>CHAPTER 8. FUTURE WORK AND CONCLUSION .....</b>	<b>57</b>
8.1. FUTURE DIRECTIONS.....	57
8.2. CONCLUSION.....	57
<b>CHAPTER 9. REFERENCES .....</b>	<b>58</b>
<b>ANNEXURE .....</b>	<b>61</b>

## List of Figures and Tables

Figure 1: Urdu Alphabet .....	10
Figure 2: Diacritics in Urdu.....	11
Figure 3: Classification of Urdu Alphabet .....	11
Figure 4: Urdu sample text written in Naskh and Nastaleeq (Right to left).....	12
Figure 5: Different writing styles for Arabic script [5] .....	12
Figure 6: Nastaleeq stacking and diagonal behavior [8] .....	13
Figure 7: Isolated characters .....	13
Figure 8: Ligatures on baseline .....	14
Figure 9: Nastaleeq joining example (Initial shapes of bay) .....	14
Figure 10: Nastaleeq joining example (Medial shapes of Bay “ب”) .....	14
Figure 11: Nastaleeq overlapping (Within ligature).....	15
Figure 12: Nastaleeq overlapping (Between ligatures) .....	15
Figure 13: Overlapping and horizontal space comparison of Nastaleeq and Naskh .....	15
Figure 14: Diacritic shifting (a) and problem in placement (b) .....	16
Figure 15: Proposed model of OCR system.....	28
Figure 16: Sample image and its thinned form .....	30
Figure 17: Thinned image and free points .....	31
Figure 18: Segmentation point.....	31
Figure 19: Junction point inspection.....	32
Figure 20: Segments of sample image .....	32
Figure 21: Perpendicular cutting.....	33
Figure 22: Segments of sample image .....	33
Figure 23: Pseudo code for segmentation algorithm.....	34
Figure 24: Framing illustration .....	35
Figure 25: 1st, 2nd, 3rd, 7th, and 9th frame of first segment.....	36
Figure 26: Sorting of segments.....	37
Figure 27: Diacritic association.....	38
Figure 28: Post Processing Unit (Unicode Mapping).....	40
Figure 29: Application to get ligatures having train and trained classes.....	42
Figure 30: Training application .....	45
Figure 31: Testing application.....	46
Figure 32: A wrong junction point was made during the thinning process .....	49
Figure 33: Segmentation error because of absence of junction point .....	49
Figure 34: The segment in the left image is misclassified with the right one .....	51
Figure 35: Ligature with its thinned image .....	54
Figure 36: Sorted segments of ligature.....	54
Figure 37: Examples of diacritic association error .....	55
Table 1: Examples of extracted ligatures.....	41
Table 2: Examples of extracted ligatures for class “ج”.....	42
Table 3: Segmentation results .....	48
Table 4: Recognition results.....	51
Table 5: Class-wise recognition results .....	52
Table 6: Post processing results.....	54



## Chapter 1.Introduction

Optical Character Recognition is a process that can convert text, present in digital image, to editable text. It allows a machine to recognize characters through optical mechanisms. The output of the OCR should ideally be same as input in formatting. The process involves some preprocessing of the image file and then acquisition of important knowledge about written text. That knowledge or data can be used to recognize characters. OCR is becoming an important part of modern research based computer applications. Especially with the advent of Unicode and support of complex scripts on personal computers, the importance of this application has increased [14].

The current study is focused on exploration of possible techniques to develop an OCR system for Urdu language. A detailed analysis of Urdu writing system has been done in order to understand the core challenges. Existing OCR systems are also studied to know the latest research going on in this field. The emphasis was on finding workable segmentation technique and diacritic handling for Urdu ligatures, and built a recognition module for these ligatures. The complete methodology is proposed to develop an OCR system for Urdu and a testing application is also made. Test results are reported and compared with the previous work done in this area.

\*\*\*\*\*

## Chapter 2. Urdu Writing System

### 2.1. Introduction of Urdu writing system

Urdu is an Indo-Aryan language and spoken by about 104 speakers all over the world [2]. It is the National language of Pakistan. Urdu is also spoken in Afghanistan, Bahrain, Bangladesh, Botswana, Fiji, Germany, Guyana, India, Malawi, Mauritius, Nepal, Norway, Oman, Qatar, Saudi Arabia, South Africa, Thailand, the UAE, the UK and Zambia [3].

Urdu is written in Arabic script with some additional characters which are not present in Arabic language. It is a bidirectional language. Words start from right side and numbers are written from left to right. This bidirectional nature of language increases the complexity of Urdu writing system.

Urdu alphabet is comprised of 38 basic characters [8]. These characters are joined together to make words of the language. The very interesting phenomenon that Urdu script exhibits is change in basic shapes of characters. This change of shape is dependent on the position of the character in ligatures. This leads to four possible positional categories in which shapes of a character can be divided. These are initial, medial and final positions of a character in a ligature and the isolated one. One character can acquire several shapes in each position. The shapes of a character are dependent on characters coming before and after it. In some cases this dependency is based on 3 following characters [8]. Some characters can only come at the final position, so they can be called ligature finishers [4].

### 2.2. Alphabet

As described earlier that Urdu character set is comprised of 38 basic shapes. It does not include 'Aerab' (Diacritics used for pronunciation and vowel sounds). This alphabet is shown in figure 1.

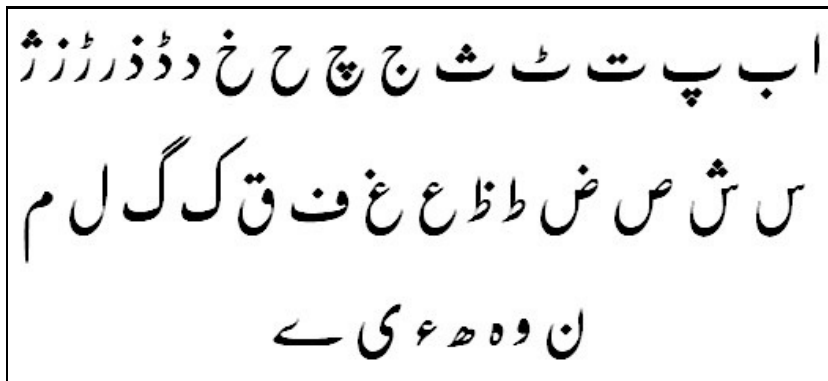


Figure 1: Urdu Alphabet

## 2.3. Diacritics

Urdu characters are enriched with some marks, called diacritics. One type of diacritics is 'Nuqta' (Dot). Dots are essential part of some characters. Their position as well as number is important in distinguishing a character. Dots are part of the characters.

Another type is called 'Aerab'. These are vowel marks and helpful in pronunciation of Urdu word. In Figure 2 common diacritics are listed. The circles represent a character to clarify the position of diacritic [8]. 'Aerab' are written in text when there is confusion in the pronunciation, otherwise they are optional and normally not written in regular text.

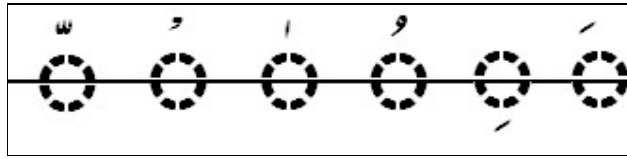


Figure 2: Diacritics in Urdu  
(Right to left: Zabr, Zair, Paish, Khari Zabr, Juzm, Shud)

## 2.4. Classification of Alphabet

In Urdu Alphabet, several characters are similar to each other with respect to their basic shapes. They vary from each other because of diacritics (dots) attached with them. Hence, we can group the similar characters to minimize the diversity. All 38 characters of Urdu alphabet can be classified into 21 classes [6]. This classification is shown in following figure.

ق	11.	آ	1.
ک گ	12.	ب پ ت ٹ ث	2.
ل	13.	ج چ ح خ	3.
م	14.	د ڈ ذ	4.
ن	15.	ر ز ژ	5.
و	16.	س ش	6.
ہ	17.	ص ض	7.
ھ	18.	ط ظ	8.
ع	19.	ع غ	9.
ی	20.	ف	10.
ے	21.		

Figure 3: Classification of Urdu Alphabet

The medial shape of ن and ی is similar with ب so they are also considered part of class 2.

## 2.5. Writing styles

Urdu can be written with different styles. Two of them, Naskh and Nastaleeq are commonly used. Naskh script is written on horizontal baseline while Nastaleeq is diagonally directed from top right to bottom left corner [5]. Nastaleeq is the conventional writing style for Urdu printings.

علامہ اقبال نے پاکستان کا خواب دیکھا۔ علامہ اقبال نے پاکستان کا خواب دیکھا۔

Figure 4: Urdu sample text written in Naskh and Nastaleeq (Right to left)

There are a few more writing styles used in Urdu calligraphy. A handwritten example of them is given below.

نستعلیق	وَسَخَّرَ الشَّمْسَ وَالْقَمَرَ
نسخ	وَسَخَّرَ الشَّمْسَ وَالْقَمَرَ
کوفی	وَسَخَّرَ الشَّمْسَ وَالْقَمَرَ
ثلث	وَسَخَّرَ الشَّمْسَ وَالْقَمَرَ
دیوانی	وَسَخَّرَ الشَّمْسَ وَالْقَمَرَ
رقاع	وَسَخَّرَ الشَّمْسَ وَالْقَمَرَ

Figure 5: Different writing styles for Arabic script [5]

## 2.6. Detailed analysis of Nastaleeq

Nastaleeq is the most popular and widely used writing script among all available Arabic scripts styles for Urdu writing. It is also considered as one of the most complex scripts to be used electronically [6]. It was evolved from two different scripts named 'Naskh' and 'Taleeq'. It was named as Naskh-Taleeq and then shortened as Nastaleeq.

Before the availability of Open Type Fonts, it was written with the help of pre-stored ligatures. But that solution was very limited in terms of usage. Users had to get one of specific software to write Urdu documents. An example of such software is Urdu Inpage.

After the advent of Open Type Fonts (OTF), a simpler solution was found to write Urdu. There are many OTF fonts available which facilitate Urdu users very well. Some of these fonts are

Nafees Nastaleeq, Nafees Naskh, Nafees Web Naskh, Nafees Rika and Nafees Tehreer Naksh [7].

### 2.6.1. Diagonality and Stacking

Nastaleeq is written diagonally at different angles with the horizontal axis. This kind of direction allows variable height and width of ligatures. Also there is no limit of number of characters to be joined together. Hence, the stacking of characters makes some ligatures higher than usual ones, and consumes less horizontal space.

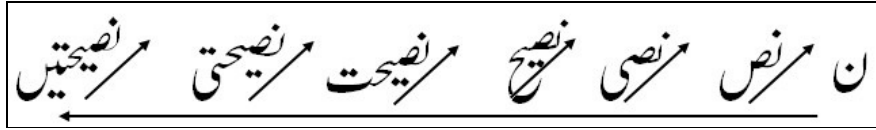


Figure 6: Nastaleeq stacking and diagonal behavior [8]

The Figure 6 illustrates diagonality and stacking of characters. The word formation is shown in seven steps. In every step one character is added. With the addition of a character, former characters are piled up diagonally and move towards the top right corner.

### 2.6.2. Hidden baseline

In spite of diagonality of Nastaleeq writing system, the written text of Urdu resides on hidden baseline. Mostly ligature starts from top right and ends bottom left. That means the text line is parallel to x-axis. This baseline is very important in writing process and different characters are present at different vertical distances from this baseline. If we take example of isolated characters, some are written in lower part of line of text and some are present in the top half. Figure 7 illustrates a line of text written of isolated characters.

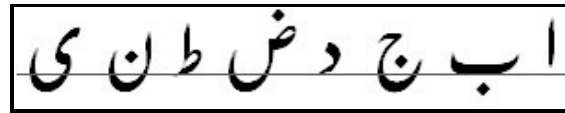


Figure 7: Isolated characters

As shown in the figure, there is a hidden baseline that is used as an aid for writing. The position of isolated characters is fixed according to that baseline in hand written text. When characters are joined the last character of the ligature should be in its actual position relative to baseline. So the pilling of character tends to be in top half of text line.

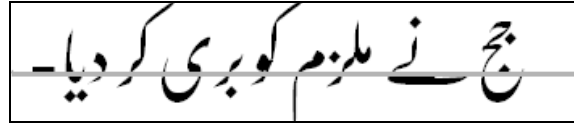


Figure 8: Ligatures on baseline

### 2.6.3. Joining and shape conversion

Nastaleeq demonstrates joining of multiple characters to form a ligature, and it affects the shapes of characters. Unlike from Naskh, where a character can only exhibit 4 possible shapes, Nastaleeq has multiple shapes of a character for a single position. Some of the characters can adapt up to 50 different shapes [8]. It means a character can have different shapes at initial, medial and final position in a ligature. The phenomenon of shape changing is dependent on addition of characters before and after that particular character.

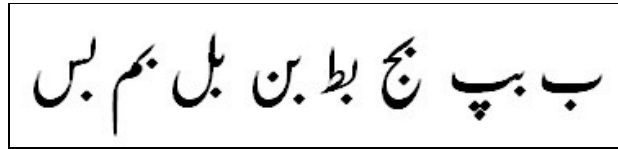


Figure 9: Nastaleeq joining example (Initial shapes of bay)

Some possible shapes of character 'ب' are shown in Figure 9. In these examples 'ب' is present at the initial position of ligatures. The following figure illustrates some medial shapes of the same character.

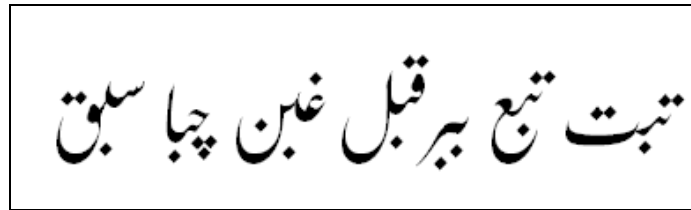


Figure 10: Nastaleeq joining example (Medial shapes of Bay "ب")

Generally, most of the characters coming at the end of a ligature adapt a shape similar to their isolated one. There are some specific classes of characters, which can only come at the ending position in a ligature. So either they come in isolated forms or as ending character of a ligature. Some of these characters acquire new shapes when joined. So it is not hard and fast that characters at the end of ligatures are always in their original shape.

#### 2.6.4. Overlapping

Due to diagonal writing style, overlapping is shown by Nastaleeq characters. One type of overlapping is within the ligature [5]. That enables certain characters to be written in reverse direction (left to right), and hence, showcases a non-monotonic behavior. Sometimes when joined, these characters end up on the right side of starting point. This phenomenon is depicted in Figure 11 where four letters 'ع', 'ب', 'ج' and 'ع' are joined to make different combinations.

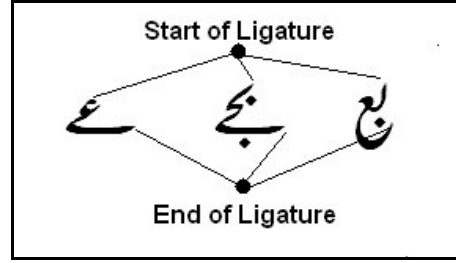


Figure 11: Nastaleeq overlapping (Within ligature)

Another type of overlapping is present in Nastaleeq writing style. It happens when some characters belonging to different ligatures overlap each other. This type of overlapping results in eliminating the vertical white space between two ligatures and making ligature isolation more complex.

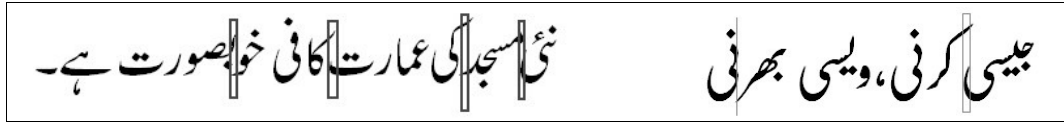


Figure 12: Nastaleeq overlapping (Between ligatures)

Both types of overlapping have significant importance in space saving for the writing style. The following figure shows both kinds of overlapping shown by Nastaleeq style and consumed space comparison with Naskh style.

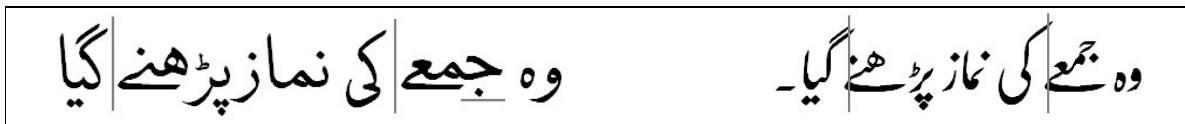
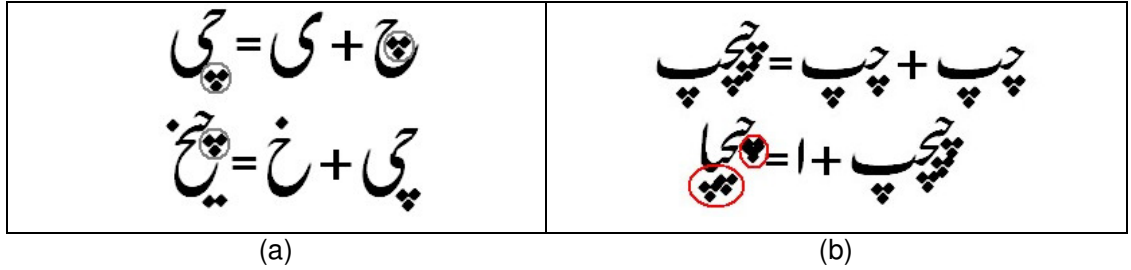


Figure 13: Overlapping and horizontal space comparison of Nastaleeq and Naskh

#### 2.6.5. Diacritic placement and shifting

Joining and overlapping features of this script makes diacritic placing quite tricky. Diacritics, especially 'Nuqta' are essential part of characters. When two or more dot holding characters

are joined together, the position of dot(s) has to be changed to accommodate all diacritics. The vertical overlapping between ligatures and ligature overlapping feature of Nastaleeq script makes this issue more complicated [5]. In the following figure different examples are shown to point out various complex scenarios.



**Figure 14: Diacritic shifting (a) and problem in placement (b)**

The example (a) describes the shifting of three dots of 'چ' when it is joined with other characters. The major issue is experienced when two or more such characters are joined which contains dots. The extreme case is illustrated in example (b) of Figure 14.

## 2.7. Noori Nastaleeq

Noori Nastaleeq is a writing style designed by a Pakistani calligrapher, Ahmed Mirza Jamil. The ligatures in this writing style were designed to be stored easily in computer. There was not formula or rule to define these writing strokes. So for more than 18,000 ligatures a database of glyph was made [36].

In early eighties, Monotype implemented Noori Nastaleeq in their LaserComp machine. Jang, one of the largest newspaper groups, bought this machine at that time to publish newspaper in Urdu. Later on, different software interfaces were made to use these pre-stored ligatures. A few of these software are Urdu Inpage, Surkhab and Shahkar [11].

\*\*\*\*\*



## **Chapter 3.Literature Review**

### **3.1. OCR System**

#### **3.1.1. Importance of OCR**

The main objective of OCR system is to import printed text into machine, and enabling user to edit it with minimal effort. But there is a lot more modern usage of this application. Some of them are listed below.

##### **3.1.1.1. Document conversion**

OCR can create editable text document with little human effort that can save space and time as well [1]. It can be used to publish the online content as text instead of images. It can help not only in saving the storage space but also a huge collection of Urdu corpora can be generated. It can aid language processing applications which requires reasonable amount of data for statistical calculations.

##### **3.1.1.2. Online search**

Modern search engines can be equipped with OCR system. Hence, user can search online images with required text in them. The significance increases with the fact that still majority of news websites use images to display Urdu text. Online content is hard to search.

##### **3.1.1.3. Text to speech**

Text to speech application can use OCR system to get the textual form of an image out of a document. It can help to enhance the applicability of TTS system. Applications developed to aid visually impaired people can use OCR system to increase their versatility.

### **3.2. Generic OCR modules**

OCR systems proposed by different researchers are comprised of various modules. Almost every one proposed a preprocessing unit to clean the image and make it ready for recognition process. In the following section, brief overview of preprocessing is given.

#### **3.2.1. Preprocessing**

##### **3.2.1.1. Skew detection and removal**

Skew in the image can be a factor for decreasing the accuracy of recognition system. The document angle with horizontal line is very important in document model acquisition. The angle of text should be same during training and recognition processes. Even slight change

of angle can make it difficult to isolate text from image correctly. The reason is that, most of the time line spacing is not very good for the text of complex scripts. A lot of marks and diacritics are attached and with the change of angle they can overlap each other during the isolation process of text lines. So its detection and removal is necessary to provide accurate input for recognition system [1, 15].

There are a few precise algorithms developed for skew detection which promise to detect skew up to certain angle. Baird's Algorithm detects skew up to 15 degrees. Hinds et al. Algorithm claims detection of skew less than 45 degrees. O' Gorman's Algorithm, Postl's Algorithm, and Le at el. Algorithm are also used for skew detection [15].

### **3.2.1.2. Noise removal**

Noise is another hindrance in acquisition of clean signal for recognition. An unprocessed image can have noise due to many reasons; quality of paper, printing intensity, scanning device's effectuality and any other environmental reason can result into noise in the image. Sometimes a particle or finger print on the scanning device can cause an unwanted mark to be present in the image. Noise is largely present at the vacant area in the pages, out of text boundaries.

Apart from accidental marks present in the image, another kind of noise may occur in the image. This noise is because of the process of conversion and called as channel noise. When images are scanned, sometimes smaller text entities are grouped together and make some new shapes. This can happen with dots in case of Urdu. This noise should also be removed before further processing [10].

Faisal Shafait with his co-researchers found a way to get the frames of text present in image and isolate the noisy data [13]. He discussed different techniques estimating text bodies, starting from novel technique of connected components and size, shape comparison. An efficient algorithm was proposed in their research to detect and remove noise. It involved text frame area detection and then finding the connected text component. At the end isolation of noise particles out of text was done, which was different from textual entities. Their test results showed approximately 97% accuracy in noise removal.

### **3.2.1.3. Document model acquisition**

Document model has an important role to play in appropriate data acquisition for recognition. Scanned document can have different type of objects present in it. It can have figures, tables, graphs, and more type of entities. In addition to that, a table can have text

present as well. Text can be comprised of different fonts, sizes and formatting. That information is needed to call the appropriate recognition module. Document model acquisition is now part of modern OCR systems and specially has great significance in multiple language recognition systems [13].

#### **3.2.1.4. Unit isolation**

After the separation of text with the help of document model, next step is isolation of rows and then extraction of basic units out of lines. These units are connected components which are used to construct the string of text (words, sentences etc.). These units are processed further, if required and then used for recognition. In most OCR systems, unit isolation is trivial and it is done with the help of white space present horizontally between lines and vertically between units. This is last step for preprocessing on images [14].

#### **3.2.2. Segmentation**

Segmentation is an important step in a lot of complex script OCR systems. Mostly the units generated as a result of preprocessing are not the basic building blocks of that particular script. So there must be an algorithm to separate smallest possible shapes which are used to construct the complex units. These smallest shapes are called segments. The extraction of these segments is necessary for the recognition module [18].

Segmentation is needed for those scripts in which the separable units of preprocessing are large in number, and many of them share same stroke sequence. OCR systems can be classified on the bases of segmentation. A detailed analysis of such system has been done in next section.

#### **3.2.3. Recognition system**

This is the core module of OCR system. Appropriate features from segments or units are extracted and passed to the recognition system and its identity is acquired. Generally recognition system is trained before hand for that data. Various recognition systems have been used to build OCR. Some of them are very trivial, while others are based on advance machine learning and artificial intelligence techniques. The accuracy of OCR is affected by recognition module's results.

#### **3.2.4. Post processing**

Post processing of OCR includes ligature formation from recognized segments and eventually production of meaningful words. The word production is a challenging task in

Urdu language, as there is no orthographic space is present between two words. Spelling and grammar checking is also part of post processing of OCR system.

The post processing unit is sometimes designed as a feedback module. On the basis of the results of post processing unit, multiple results from recognition system can be considered as possible outputs. So, post processing unit can provide feedback

### **3.3. Historical background (*Current State of OCR Systems*)**

#### **3.3.1. English OCR**

Research on English OCR systems have been carried out since last 2 decades of 20th century [14]. During last a few years, there has been some robust systems emerged in the market which provide recognition functionality with high accuracy rates. The major technique used by these OCR systems is pattern recognition for classification of different Latin characters.

In 2002, Jörg Schulenburg proposed an OCR technique based on feature extraction. The technique emphasized on spatial feature extraction from top to bottom of text lines. After that distance measuring is used. Euclidian distance between training and testing features is used to complete the recognition process. Rule based engine is merged with this OCR to enhance the results as a post processing engine. This system is released under the name of **GOCR** system. Recently its version 0.46 is released which can detect pictures from the image and ignores them. It extracts text and dispatches it for processing to the core engine. This latest version of GOCR was released in 2008 and improvement is ongoing for this project [19].

**Ocrad** OCR is a GNU product and comes with Linux operating system. It recognizes English strings on relatively clean background. It uses the features extraction and matching techniques to generate UTF8 text. The preprocessing module of this OCR is comprised of layout analyzer, which extracts text strings from a document containing shapes and tables [31].

Hewlett Packard (HP) started an OCR research project named Tesseract in 1984. The developed system was capable of recognizing English text strings with good accuracy measures. The research on this project was carried out until 1995, after that it remained dormant but later it became active again. In 2005 HP released Tesseract under public license. Google is now working on this system as an open source project.

**Tesseract** recognition process starts with line separation and baseline identification. Baseline helps in separating the characters from each other. Isolated characters are eventually sent to core recognition engine which is currently working on pattern matching technique. The features from the characters are extracted in the form of polygon approximation [23]. On the basis of these features, shape recognition is performed in two steps. In the initial step, list of all probable shapes is created which can be the match for test shape. Feature matching is used for listing down these shapes. In second step, distance matching is used to identify the correct classification of test image. Current accuracy of Tesseract for different types of documents is reported nearly 99% [37].

Tesseract is used as recognition engine in **OCRopus** system which is module based document analysis & multilingual OCR framework sponsored by Google. There are two different recognition engines used so far for this framework. Tesseract is already discussed in details and the other one uses neural networks for recognition of characters. This system is still under progress and is not as much accurate as Tesseract [20].

### **3.3.2. Arabic OCR**

Arabic OCR problem is very much similar to Urdu because of same script. Arabic is written in Naskh writing style, which shows less shape changing phenomenon than Nastaleeq. It is written along a horizontal baseline with clear horizontal coverage of characters within a ligature. In spite of these facts segmentation of Arabic characters from ligatures has been a challenging task for researchers [27].

A working solution for segmentation and recognition was proposed by Khorsheed et.al in 2000 [24]. They worked on gathering spatial features of Arabic words and stored them as polar coordinates to handle the skew of the words. Complex numbers were used to demonstrate these features, and hence, the computation cost was reduced by converting these features to Fourier transform. The transformation also helped to reduce the number of features. Template matching was used in that experiment to recognize words. After training the system, they tested it on 1700 words of different fonts and achieved an accuracy of 98 % [24].

In 2001, a graph based segmentation technique was proposed by Elgammal and Ismail [25]. Their work began with finding the baseline for Arabic text string. Horizontal projection profile was used to estimate baseline with minimal error. Highest number of pixels in a row was an obvious hint of baseline. After finding baseline, the technique focused on sub words

and script extraction<sup>1</sup>. Line adjacency graph (LAG) was used to find the connected strokes. Node minimization technique was applied on constructed LAGS and its descendants were called c-LAGs. c-LAGs were classified with respect to baseline. Finally Bayesian classifier was used to cluster the segments with training set. Separate classification units were used for main bodies and diacritics. The experimental results showed an accuracy of 94.6 %.

Hand writing recognition is considered as more error prone system than OCR because of its input variations and hence, should be more robust as well. An offline hand writing recognition system was proposed in 2002 by Alma'adeed and his colleagues [26]. They used HMM classifier to identify the handwritten input. The first task was to determine the correct angle with baseline. Then connected strokes were isolated. The spatial features are extracted from each connected body and dispatched to HMM classifier. The reported accuracy is 45% without training of a user and it elevated up to 78% when training was done.

ORAN system was proposed in 2004 by Zidouri [27]. The basic algorithm of the system addressed shape changing issue in Arabic writing style. A classification of initial, medial final and isolated shapes of Arabic alphabet was made. Then the isolation of recognizable units was carried out. Minimum Covering Run (MCR) length was the term, they used for a connected component body. Different patterns of MCR were saved in a database and eventually used for recognition. Various features were extracted including coordinates of an MCR, position with respect to baseline, direction of stroke and number of strokes. The test results of 6000 characters showed an accuracy of 97 %.

In 2006 another handwriting recognition system was proposed by a group of researchers for Arabic and Chinese writing [29]. The segmentation is simply gap based technique; finding white spaces between connected strokes. A graph based method is used to separate spatial features of strokes. Pattern matching was used for recognition purpose. The unit for recognition must not be a word in this system. Sub words and ligatures were also trained and tested. The same algorithm was applied on Chinese hand writing as well. The segmentation also worked on this language. Test results showed accuracy of 85 % when it is writer dependent.

In 2007, different Arabic OCR systems were compared in ICDAR conference [30]. The most accurate system presented there was an adaptive Latin based OCR system. The preprocessing of Arabic writing was done as a regular procedure. First of all baseline

---

<sup>1</sup> Script was a term used by Elgammal et. al. for ligature segments

estimation was done, then feature extraction and eventually a mapping algorithm for Arabic to Latin conversion. HMM was used as recognizer in that system and spatial features were used for training. Models for each character were trained and then tested for a reasonable text image corpus. The results presented in conference show 94 % accuracy different samples of text which included 4,000 words.

### **3.3.3. Urdu OCR**

Development of OCR system for Urdu language has remained a challenging task for Urdu researchers during last few years. The main reason for its challenging nature is complexity of Urdu writing system. As discussed earlier Urdu is widely printed in Nastaleeq writing style. A detailed survey carried out by CRULP showed that almost all of the books (Printed Text) are published in Noori Nastaleeq font. The complex nature of Nastaleeq was analyzed in Section 2.6. The cursive and context based shape changing phenomenon of Urdu makes the isolation of characters harder as compared to Latin script. So segmentation itself is proven to be an error prone and difficult task. Hence, most of the developed systems are based on ligature recognition methodology.

A segmentation free approach was used by Hussain in which ligature segmentation was not involved [16]. The main bodies were extracted from diacritics and then statistical features were acquired. These identified features were used to recognize the ligature through a multi-tier neural network model. The trained system showed accuracy higher than 95% on test samples of two character ligatures.

In 2005, same feature extraction technique [17] was used to develop a ligature based recognition system. Spatial features were extracted from the isolated ligatures of Urdu text and then Hidden Markov Model was trained for the recognition process. Nearly 1500 ligatures were trained and the system showed accuracy above 92% [1]. The limitation of this system was shape dependence, which means only trained ligatures were recognized by that OCR. To cover current corpus of language, all ligatures have to be trained to the system with high accuracy which itself is a tedious work. In order to cut down the training work, segmentation was necessary to encounter the shape changing phenomenon of Urdu.

In 2007, research [22] was extended to develop a segmentation based Noori Nastaleeq OCR system. It was the first attempt to divide ligatures of Nastaleeq into smaller pieces for recognition purpose. The proposed technique has following major modules.

- Line Separation
- Ligature and Diacritic Separation
- Thinning
- Segmentation
- Framing
- Recognition
- Post Processing

This system was trained and tested with 60 different ligatures, made by combinations of 6 Urdu classes (Alif, Bay, Dal, Swad, Ain, and Choti Ye). The research showed very promising results by using workable solutions for Urdu OCR. The system is discussed here in details.

The line separation process and baseline detection has been done by finding the vertical histogram of image. Clear white spaces are present between two text lines in an image. Vertical histogram shows a band of zero black pixel values and text line is separated from there. After separation of lines, the maximum value of vertical histogram is marked as the baseline for text. Instead of using single pixel baseline, a band of 5-10 pixels is chosen to fulfill the purpose.

The baseline identification was required to isolate main bodies of ligatures from their diacritic marks. All of the connected components present on baseline are considered as main bodies and rest of them are marked as diacritics. In a more recent research it has been experimented that the method is error prone because some diacritics reside on baseline as well. The proposed solution of this problem is setting a threshold on size of connected body to distinguish between diacritics and main bodies [34].

After identification of main bodies, next step in the given research is segmentation. The technique of segmentation is based on single pixel stroke so thinning of main bodies is introduced. The detail of thinning process is not elaborated in the research but it was done with the help of Matlab<sup>2</sup>. The discussed segmentation technique works with stroke cutting at the junction point i.e. during the traversal of thinned body, when more than one neighboring pixel is found a segment is separated. Some problem of under and over segmentation of this technique has also been highlighted by the researchers of this study.

As soon as segments are acquired, the next proposed step was performing framing operation to divide the segment in equal windows. A frame window of 8\*8 pixels has been

---

<sup>2</sup> An application which provides image processing tools



used for division of segments. A static disjoint framing technique is chosen and thin stroke is kept in the middle of 8\*8 sized window. Frames are numbered with the order of segment stroke sequence. After filling all frames DCT is applied and output is stored in a single file.

The system was trained to recognize only classes which are used to construct a particular ligature. Character recognition was not performed in this research. 6 different classes (ا ب د ع)

س ی) were used to construct different ligatures and system was trained for those ligatures.

Segmentation yielded 60 unique segments and HMM model was developed. A rule based post processing unit was developed to construct the ligature. Test results of all classes have been reported in the given study. Almost all samples of ا and د have been recognized

showing accuracy of 99%. Classes of ع and ب shows nearly 95% accuracy on the other hand س showed 80% accuracy [22].

The research discussed above shows very encouraging results and a devised method for segmentation based Urdu OCR system. One of the challenges described by the researcher of that study is lengthy training procedure. Grouping of segments, and manual training took very long time for building HMM model. Also the absence of sufficient features of thinned stroke was pointed out.

\*\*\*\*\*

## Chapter 4. Problem Statement

Urdu OCR system is required to convert numerous printed books of Urdu into editable computer text files. Recent research in this area has been able to evolve some new methodologies to overcome the complexity of Urdu writing style. Still these techniques have not been tested for complete characters of Urdu Alphabet. Hence, a system is required which can handle all classes of Urdu text and identify characters among these classes. Also a complete analysis of segmentation technique is needed to look into the possibilities of developing a segmentation based Urdu OCR. Literature review showed that diacritics have not been studied in Urdu OCR systems, which play an important role for correct character recognition. Hence, a hybrid technique of segmentation and diacritic association is required.

### 4.1. Scope

The scope of this study is to build an Urdu OCR system with following limitations.

- Text written with Noori Nastaleeq font in 36 point size, scanned at 300 DPI will be recognized.
- System will be able to recognize valid combinations of all characters of Urdu Alphabet given in Figure 1.
- Dots associated with ligatures will be taken care off but Urdu numerals [۰، ۱، ۲، ۳، ۴، ۵، ۶، ۷، ۸، ۹]، Aerabs [۰̣، ۰̣̣، ۰̣̣̣، ۰̣̣̣̣، ۰̣̣̣̣̣، ۰̣̣̣̣̣̣، ۰̣̣̣̣̣̣̣]، punctuation marks and symbols will not be

handled.

- The input image is assumed to be without any noise and skew.
- Single ligature based image will be accepted by the system. Document analysis will not be done by system.
- The output of the system will be in form of a string containing Unicode values of single ligature. Word level post processing will not be done.

\*\*\*\*\*

## Chapter 5. Proposed Methodology

The adopted methodology in this research study is inspired with the previous work done on Noori Nastaleeq OCR system which is discussed in literature review [22]. All modules have been developed again with suitable modifications. The error analysis of previous study showed that thin stroked segments can confuse with each other because of very little change in the shape. That is why original stroke of the segment is captured to enhance the recognition results. A module is added to capture thick segments. In addition to that, diacritic recognition is carried out and a post processing unit is developed to construct ligatures with original characters instead of classes.

The modified design model is given in the next subsection. All the modules are discussed in details, along with the modification in older technique. The system is programmed with incremental development model, and comparative testing is being done whenever there were two options to follow. Challenges faced during development of these modules are also discussed for the ease of future work and reference.

### 5.1. Design model

Proposed system for Urdu OCR has been designed with the help of various modules. Every logical step is implemented with separate module and output of each module is explicitly stored for analysis. The testing application is made by dispatching output of one module to other and finally result is obtained. The input of this model is cleaned single ligature and output is Unicode string of that ligature. Following design model is used for development of this system.

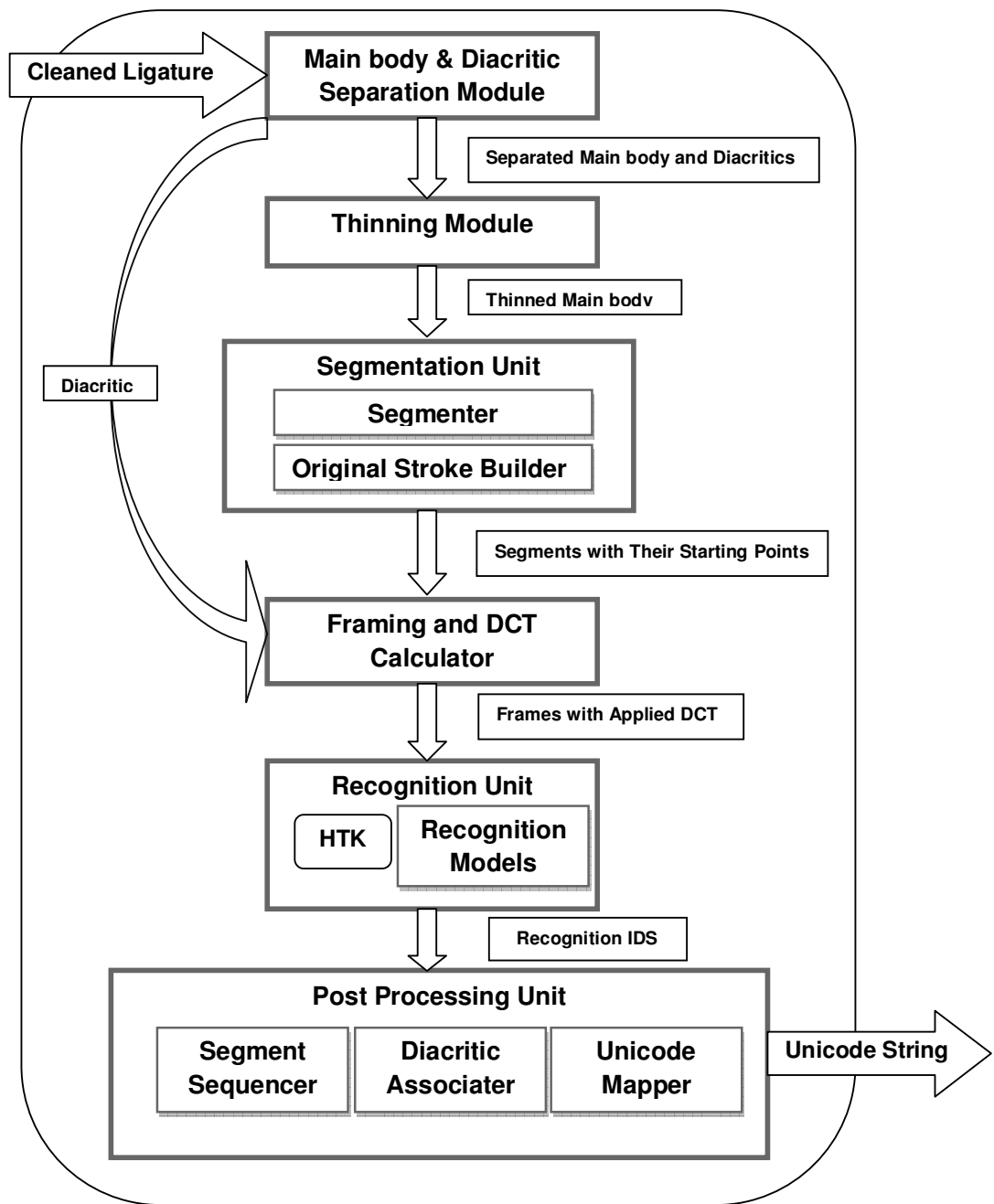


Figure 15: Proposed model of OCR system

## 5.2. Main body and diacritics isolation

The first step was to isolate main bodies of ligatures. In this research ligatures are referred as a complete stroke of connected characters (Main body) including their diacritical marks. Main body is isolated from a ligature on the base of size as it is longest connected component of a ligature. Isolated main body is stored in separate file. All other marks are captured as diacritics. The main bodies of a ligature depict the classes which are involved in the construction of that ligature and diacritics are used to identify characters of these classes.

## 5.3. Thinning (Skeletonization)

### 5.3.1. Process definition

Thinning or skeletonization is a process of converting thick main body of ligature into a thin line stroke. It is an image processing technique which helps in extracting the skeleton of an image. It has been widely used in OCR projects for cursive scripts [29]. The resultant image is single pixel thin stroke of all connected components present in given file.

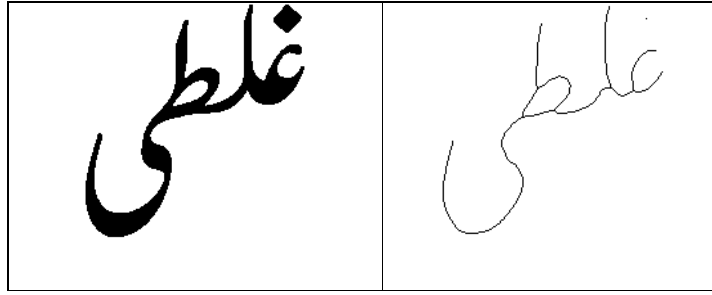
### 5.3.2. Process details

Thinning is required for the particular segmentation technique used in this study. Segmentation of main body was required to find the building block of a ligature which can be a complete character, partial character or combination of some characters<sup>3</sup>. Thinning process was essential to perform segmentation. In previous research, thinning was done with the help of Matlab. Although it was working fine but to remove the dependency of Matlab, a separate algorithm was implemented.

Thinning is applied on whole ligature file but segmentation is done for main body only. The thinning algorithm used in our study was developed by Jang Chin [35]. There are 31 different masks used in a sequence to find the middle stroke of connected body. After masking, we achieved the middle stroke of main body which is a single pixel line. After development, thinning module was tested for 100 unique ligatures of size defined in scope. The test results showed that all ligatures were converted into single pixel stroke. There was not any breakage of connected component. In Figure 16 one ligature and its thinned form is shown.

---

<sup>3</sup> Definition of a segment in this study



**Figure 16: Sample image and its thinned form**

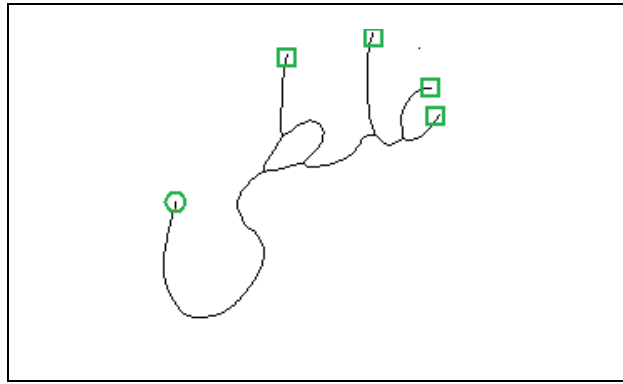
## **5.4. Segmentation**

The segmentation strategy has four basic steps including starting point detection, segmentation point detection, segmentation and restoration of original segment. The segmentation algorithm is same as used in previous research. The main difference in current research is restoration of original segment data. Starting point detection is also different from previous work [22]. These sub modules are discussed in details in the following sections.

### **5.4.1. Starting point**

We have got the thinned image of main body as input in this module. In order to find out segmentation points in the thinned main body, first step is to find the starting point of the ligature. Initially all free pixels of the ligature are found. A pixel is called free which is joined with one neighboring pixel. The proposed starting point is the left-most free pixel of the ligature which is encountered from bottom to top traversal. In most of the ligatures this point is the ending pixel of a thinned Urdu ligature as Urdu is written from right to left. Although in this technique, the recognition sequence is not dependent on selection of starting point, but for the consistency of segmentation always left most pixel is chosen.

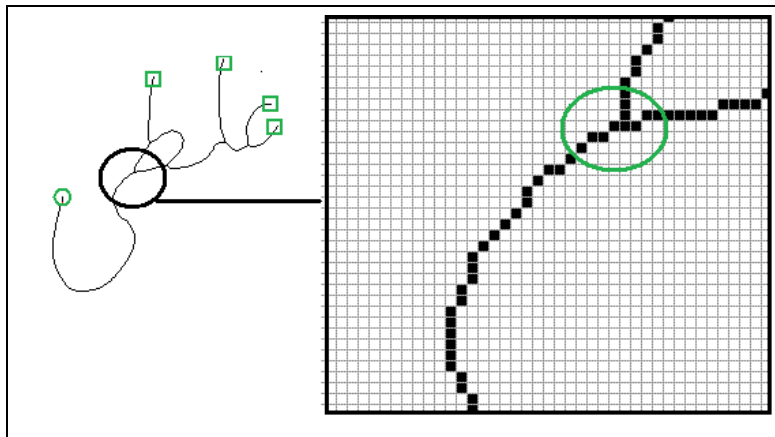
The left-bottom free point is always chosen to maintain consistency for the starting point. It makes sure that same main bodies are always traversed from same starting point. The sequence of stroke is very important for the construction of segmented chunks of main body. Figure 17 shows all free points in a thinned main body and starting point is highlighted with a circle.



**Figure 17: Thinned image and free points**

#### 5.4.2. Segmentation procedure

After finding the starting point, traversal of thinned image is carried out. We chose the chain code algorithm for traversing the thinned stroke. In simple words, starting from the initial free point thin stroke is followed. Whenever two paths are found (junction) the traversed portion is declared as a segment and traversal of next 2 paths starts from there.

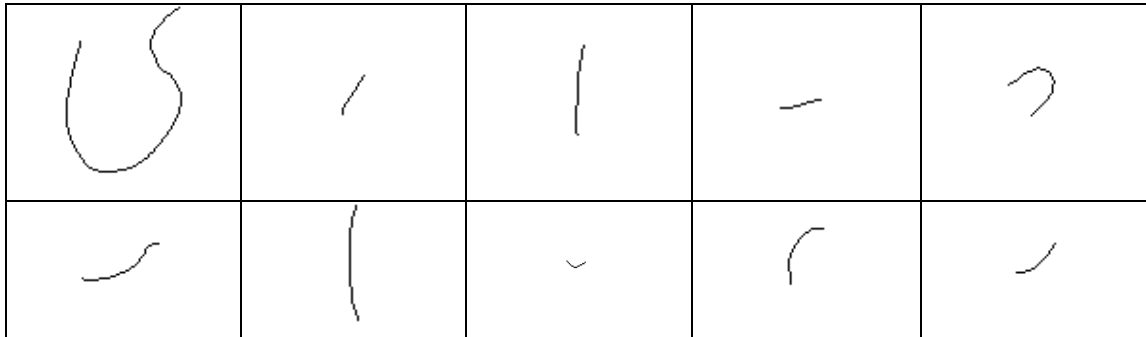


**Figure 18: Segmentation point**

All traversed pixels are marked with a flag and hence, traversed only one time. The segmentation algorithm declares sequence of pixels as one segment whenever it reaches either junction point (2 paths) or a free point (dead end). The algorithm traverses all points of thin strokes and captures all segments used in the construction of main body of that ligature. These thinned segments along with their starting points are stored in separate files. Unlike previous work, there is not any ordering of segments at this point. Whenever a junction point is encountered, all next points are sent to stack in clock wise position. The following figure explains the working. At any point all neighboring points are checked from left bottom. If there is more than one point found then all of them are pushed into stack starting from 0-7 index. So clock wise insertion is done in the stack. Segments are shown in Figure 20 as well.

2	3	4
1	<b>P</b>	5
0	8	6

**Figure 19: Junction point inspection**



**Figure 20: Segments of sample image**

### 5.4.3. Restoration of original segment

The thinned segments contain very less number of black pixels which can make it difficult to be differentiated from other segments. This problem was faced during the earlier research carried out on Nastaleeq. As a solution, original shape of the segment from its ligature has been extracted to enhance the number of spatial features for recognition system.

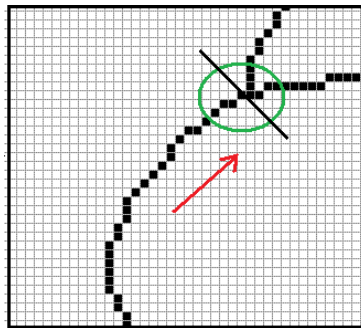
#### 5.4.3.1. Filling

The process is carried out by capturing the neighboring pixels of thinned stroke. Whenever the segment is completely found in the main body, its actual shape is also retrieved as much as possible using connected component technique. Masking has been the only option to get the connected black pixels. Two different mask sizes have been tried. Initially 8\*8 sized mask was used. But it cause trouble at the junction point. The window size was too large for the joints. Hence, 5\*5 sized mask has been chosen. The mask size has been decided after considering the thickness of the stroke. Although there is some information loss when the segments are short in length but eventually the acquired shape is very similar to the original stroke. All the thin segments less than 5 pixels are ignored (as 8<sup>th</sup> segment in example ligature).



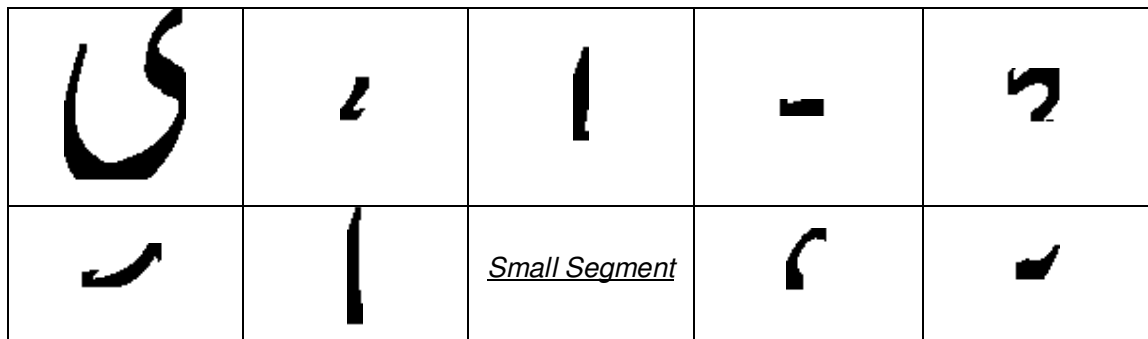
### 5.4.3.2. Cutting

The extraction of original segment is difficult at the joining ends of three segments. At the junction point, at least three segments are connected to each other. The precise cutting is required to separate them. Study of thin and thick segments showed that at segmentation point, ligature can be cut at the angle of 90° to isolate all segments. This angle is measured with the thin stroke. In other word during the traversal of thin stroke when junction point is found, it is sliced perpendicular to the traversed stroke, as shown in Figure 21.



**Figure 21: Perpendicular cutting**

Similarly, all other generated segments are also sledged with same method. Finally the isolated segments are stored in separate files with their starting points. The following figures show original segments.



**Figure 22: Segments of sample image**

### 5.4.3.3. Challenges

There have been some problems faced during the restoration of original stroke. Although cutting is solved with perpendicular slicing but for the smaller segments it is not very suitable. Very small cusps like structures are present in Nastaleeq style. The thin stroke encounters so many junction points that it becomes hard to find the perpendicular line. In

such cases filling is narrowed down near segmentation points. Majority of instances are faced in instances of class “س”.

Another problem faced during filling process is over segmentation within some portion of very thick stroke. If thinning produces unwanted branches inside the thick stroke (As in the head of “و”) then over segmentation is carried out. During filling process some of the segments are engulfed by other filled segments. This leads to construction of two segments with same black points. The problem was solved by changing the color of used black pixels. That means, when some part of the original stroke is consumed its value is changed so that it cannot be used again for filling purpose. Resultantly, very tiny unwanted inner segments are left out and not produced.

#### 5.4.4. Segmentation algorithm

The algorithm for segmentation can be described in following steps.

```
Push initial pixel into stack (Main body starting point)

Loop until stack is not empty
{
    Pull stack
    Start chain code and traverse the stroke
    If junction is found then
        Push next 2 points in stack and
        Mark the segment
    Else if dead end is found
        Mark the segment
    Else if traversed all points
        Mark the segment and exit the loop
} // Loop Boundary
```

Figure 23: Pseudo code for segmentation algorithm

### 5.5. Framing & DCT

The next step in process is preparation of units for HMM toolkit. The need of framing and DCT is discussed in next section.

#### 5.5.1. Framing need and selected methodology

The segments of main bodies are the building blocks of characters and used for recognition process. We acquired various sized segments because of different shapes of characters. Now

it was required to divide them into small pieces of same size so that the recognition engine can be developed. This task is completed with the help of framing.

There were two options of framing available i.e. disjoint framing and overlapped framing. In overlapped framing technique two adjacent frames share some same pixels among them. While disjoint framing, as name suggests, is carried out without any sharing of pixels. In previous research disjoint framing was used [1, 22].

There are different pros and cons of both techniques. We were interested in choosing more beneficial technique in terms of recognition. It was decided to compare available options to choose which way to go. Disjoint and Overlapped framing algorithms were implemented and tested for varying sized segments of 25 ligatures. It was found that the recognition results for large segments are almost equal with both strategies. But for the smaller segments overlapped framing gave better results. The clear reason for this behavior was provision of more number of frames and hence, probabilistic model worked better for recognition.

We chose overlapping framing because of its nature of preserving contextual information. The other reason for this selection was to have some decent number of frames for each (even smaller) segment. The frame size has been decided 15\*15 to cover the width of stroke. Two consecutive frames share 7 pixels (thin stroke) among each other, that means every new frame takes 7 old pixels and rest of them are new.

### 5.5.2. Framing process

The initial points of each segment have been acquired for the framing process. The first frame contains 7 pixels from thin stroke of segment and original stroke is also filled in the frame. That means the framing is carried on thinned stroke and respective data from thick version is acquired to fill the frame. The next frame gets last 7 points as initial sequence and rest of the new points are filled along with thick segment sequence. Complete segment stroke is divided into frames. This is elaborated in the following figures.

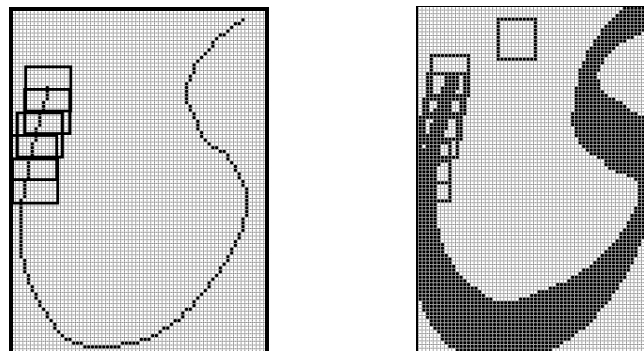


Figure 24: Framing illustration

13:2 TO 27:16	21:0 TO 35:14	29:-2 TO 43:12	61:-4 TO 75:10	77:4 TO 91:18
0000000000000000	00000001111000	00000011111000	00001111100000	11111111110000
0000000000000000	00000001111000	00000011111000	00001111100000	11111111111100
0000000000000000	00000011111000	00000011110000	00001111100000	11111111111110
0000000000000000	00000011111000	00000011110000	00001111100000	11111111111111
0000000000000000	00000011111000	00000111110000	00001111100000	11111111111111
0000000000000000	00000111110000	00000111110000	00001111100000	11111111111111
0000001111000000	00000111110000	00000111110000	00001111100000	11111111111111
0000001111000000	00000111110000	00000111110000	00001111100000	11111111111111
0000001111000000	00000111110000	00000111110000	00001111110000	11111111111111
0000001111000000	00000111110000	00000111110000	00001111111000	11111111111111
0000001111000000	00000111110000	00000111110000	00001111111100	11111111111111
0000001111000000	00000111110000	00000111110000	00001111111110	11111111111111
0000011111000000	00000111110000	00000111110000	00001111111110	11111111111111
0000011111000000	00000111110000	00000111110000	00001111111111	01111111111111
0000011111000000	00000111110000	00000111110000	00001111111111	01111111111111
0000111111000000	00001111110000	00001111110000	00001111111111	00111111111111
0000111100000000	00001111000000	00001111000000	00001111111111	00000000000000

**Figure 25: 1st, 2nd, 3rd, 7th, and 9th frame of first segment**

### 5.5.3. DCT

Discrete Cosine Transformation is applied on all frames to convert the spatial information into frequency domain. Although after applying this transformation the number of pixels used in a recognition unit (frame) is not reduced but the coherency of points in frames is strengthened because of the nature of two-dimensional transformation. All of the frames of one segment are stored in a single file after applying DCT along with some headers. This single file of the segment is prepared for the training of that particular segment. This process is same as used in previous research without changing anything except the data header. Now we are using 225 data points per frame in contrast to 64 points used earlier [22]. A separate module is developed to calculate DCT of frames.

### 5.6. Recognition

The next step is recognition of all the segments constituting the main body of a ligature. Hidden Markov Models are used as recognition engine in this study. A model has been trained (training details are given in next section) to find out the IDs of segments. The processed files of all the segments of one ligature are sent to this model and their respective identification numbers are got in return. These IDs are useful in identifying the character classes of main body.

A separate small HMM model has also been made for diacritic identification. The DCT file of dots and marks is sent one by one to this second trained model and eventually diacritics are also identified. HMMs are operated by using HMM Toolkit (HTK) which is freely available for research purposes.

## 5.7. Ligature construction (Post processing)

As soon as segments identification numbers are received from recognition module, the final stage of OCR system is initiated. There are three main parts of this stage. First of all segments are arranged in order, their respective diacritics are arranged and finally Unicode values are acquired. All three sub modules are developed in current work. In the previous study Unicode mapping was made for class level. These processes are elaborated in following sections.

### 5.7.1. Segment reordering

During segmentation process, there was not any criterion of sorting them with respect to their spatial position. They were stored according to the traversal of thin stroke. Now the order is required to find the sequence of characters. Boundaries of all segments are indentified and then arranged from left to right and down to top. By doing so, ligature is constructed in reverse order i.e. The segment appearing in bottom left is placed at the start of sequence and then the adjacent right one is place next and so on so forth. As the ligature is going up with respect to baseline (Considering reverse order), the lower segment is given preference over higher segment in ordering algorithm. The ordering procedure is depicted in following figure.













No.	Segment	Top	Bottom	Left	Right
		84	175	<u>3</u>	67
2		63	87	<u>63</u>	78
3		79	87	<u>65</u>	90
4		14	65	<u>73</u>	81
5		53	82	<u>75</u>	103
6		62	85	<u>88</u>	132
7		1	65	<u>125</u>	133
8		34	67	<u>145</u>	167
9		50	69	<u>148</u>	172

Figure 26: Sorting of segments

### 5.7.2. Diacritic association

After ordering of segments, diacritics are associated with respective segments. It has been very complicated task due to repositioning of dots and marks in Nataleeq writing style. First of all coordinates of dots are stored with respect to original image. Then adjacencies of dots are found. If two double dots are present very close to each other a margin value (5 pixels) is assigned to them on horizontal scale. That value shows that they are separated by some pixels. After finding adjacencies, their coordinated are adjusted for correct association.

The association of dots is based on two parameters. First one is horizontal overlapping of dots and some segment, and second parameter is direct distance between centers of diacritical mark and segments. Most of the time one diacritic is horizontally occupied with one segment. When some confusion is present, distance is used to find out the correct association of diacritics.

Original	Segments & dot	Top	Bottom	Left	Right
		2	23	<u>148</u>	<u>169</u>
		34	67	<u>145</u>	<u>167</u>
		50	69	<u>148</u>	<u>172</u>











**Figure 27: Diacritic association**

The shapes and data in Figure 27 show the horizontal coverage of diacritic with segments. As there is no other dot present so no adjustment has been made. The given dot is 100% occupied with the last segment and 89.90% occupied with upper segment. So this diacritic is associated with 2<sup>nd</sup> segment. In case of more than one segments claim the dot with 90% coverage then vertical distancing is measured.

### 5.7.3. Unicode mapping

The final step of post processing module is acquiring the Unicode of these outcomes. One character can have multiple segments (over segmentation) and also some diacritic association. On the other hand some characters can combine to construct one segment (under segmentation). One character class can have different number/types of diacritics to distinguish among characters. So a state machine has been developed to cope with all possible combinations.

All the possible sequences of segments and diacritics are generated and stored in a file. One sequence of states exhausts inputs and returns the Unicode value of one character. When a ligature is tested, one long sequence for all segments along with diacritics is generated and then it is found in the state table. If any fulfilling sequence is found then value is acquired, otherwise one state from the sequence is dropped and again searched in the table. So it is the longest sequence matching algorithm. For example the ligature has 4 segments having IDs 23, 45, 133 and 241. A diacritic with ID 1 is attached above the segment 241. Then the state sequence is generate as “23 45 133 241(+1)” where parenthesis indicates the diacritic and + sign means it is above 241. Now this is passed to state table. If it is not found then “23 45 133” is sent. Let us assume finally “23 45” is accepted and a value is returned. Now “133 241(+1)” is dispatched, and eventually it is exhausted as well. The real example is given in the following figure.

No.	Segment	Recognition ID
1		50
2		205
3		3
4		166
5		170
6		96
7		166
8		8
9		7
D1		0

#### State Transition Table

...  
 ...  
 .....  
 227U06A9  
 227(+4)U06AF  
 .....  
 56(+4)U0626  
 50U06CC  
 41 19 47U0635  
 .....  
 205 3 166 170U0637  
 149(-1 +5)U1008  
 96 166U0644  
 8 7(+0)U063A

Post Processor State Sequence		
50 205 3 166 170 96 166 8 7(+0)		[Not found]
50 205 3 166 170 96 166 8 7		[Not found]
50 205 3 166 170 96 166 8		[Not found]
50 205 3 166 170 96 166		[Not found]
50 205 3 166 170 96		[Not found]
50 205 3 166 170		[Not found]
50 205 3 166		[Not found]
50 205		[Not found]
50	[Found U06CC]	
205 3 166 170 96 166 8 7(+0)		[Not found]
205 3 166 170 96 166 8 7		[Not found]
205 3 166 170 96 166 8		[Not found]
205 3 166 170 96 166		[Not found]
205 3 166 170 96		[Not found]
205 3 166 170	[Found U0637]	
96 166 8 7(+0)		[Not found]
96 166 8 7		[Not found]
96 166 8		[Not found]
96 166	[Found U0644]	
8 7(+0)	[Found U063A]	

**Figure 28: Post Processing Unit (Unicode Mapping)**

There are three diagrams in above figure. The left table has all segments and diacritics with their respective recognition IDs. All segments are in ordered form and dot is attached with its segment. Now a state sequence for all segments & dots is generated. The longest sequence is looked up in state table. As it is not found then last state is dropped and again searched. Eventually it is matched with a single state sequence generating “ج”. When a sequence is matched then again longest sequence is generated without selected sequence. So now first segment state is dropped and remaining states are used to make new sequence. This time 4 states combine to make “ط”. After that 2 states make “ل” and finally 2 states and one diacritic mark combine to form “ع”. Unicode mapping module attaches these Unicode values in reverse order and generates ligature string which is output of the system.

\*\*\*\*\*



## Chapter 6. Training & Testing Methodology

Training of recognition engine has been carried out by populating the training data. There are two different models trained. One for main body segments and other model is for diacritic recognition. The training data selection and training procedure is elaborated in next subsections.

### 6.1. Training data

Training data selection has been carried out according to the usage of Urdu ligatures. The highest frequency ligatures have been taken out of the corpus. The threshold limit was 200+ occurrences in Urdu corpus<sup>4</sup>. Approximately 9000 ligatures were extracted from corpus but these were not unique considering their main body shapes. The training has been done on incremental basis. Initially ligatures comprised of 6 classes have been taken to test the developed system.

These are the same classes (اب د ع ص ی) used in earlier research [22]. After completion of this training, remaining classes are added one by one and system is trained. The detailed process of training, clustering of segments and developed helping applications are discussed in detail here.

### 6.2. Training procedure

Complete training process is divided into 5 different sub modules. The process is initially tried for 6 classes mentioned above and then one by one addition of classes has been done.

#### 6.2.1. Ligature extraction

This module is developed to extract required ligatures from Urdu corpus. A small helping application has been made to deal with corpus. At first step all such ligatures are extracted which have been made by combination of first six classes (We call them train classes). All of the characters used in extracted ligature must belong to the train classes. The correct and incorrect examples are given below.

**Table 1: Examples of extracted ligatures**

Train Classes	اب د ع ص ی
Correct Ligatures	بد، غضب، پا، ضعی، تغا، عا، غا
Incorrect Ligatures	بل، صر

<sup>4</sup> Cleaned CRULP Urdu corpus of 46 K word was used for extraction of data

The table given above is only for the first pass of extraction for initial 6 classes. After that when a new class is added for training, the extracted ligature must meet the following criteria.

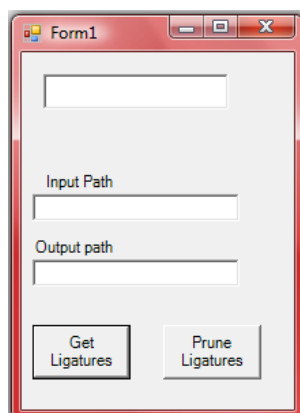
- It should have at least one character of new class.
- All other characters in extracted ligature must belong to already trained classes.

After initial 6 classes, “ج” class has been trained. All the ligatures selected in this step fulfilled the criteria given above. The examples of correct and incorrect selection are highlighted in the following table.

**Table 2: Examples of extracted ligatures for class “ج”**

Trained Classes	اب د ع ص ی
Train Class	ج
Correct Ligatures	جد، جمی، جتا
Incorrect Ligatures	غضب، جر

The table shows that train class is present in all extracted ligatures. And if a character in any ligature does not belong to already trained classes or train class then ligature is not extracted. Ligatures of all other remaining classes have been extracted in the same manner as “ج” is done. The helping application for this process is shown in the Figure 29.



**Figure 29: Application to get ligatures having train and trained classes**

In this application (Figure 29), the first text box requires a string of characters on which a search operation of corpus is performed. Corpus input path is given in a text box. All ligatures which contain characters from this string are extracted and stored in the output file in given path.

### **6.2.2. Unique ligature selection**

All the ligatures acquired in previous step are unique but not in terms of their main body. For example the selected ligatures may have “ﻝ” and “ﻝﻝ”. These two ligatures are unique but the main body is constructed with same classes. In other words the extracted ligatures are unique in terms of characters but not in terms of classes. We are doing class based training so only one of these ligatures is required to be included in training data. But we need to verify if the main body shape is same or not. Because sometimes in Noori Nastaleeq, ligatures with same classes can have different shapes. In order to select one shape, all of extracted ligatures are printed and then manually pruned. It helps in eliminating the redundant data for training.

### **6.2.3. Sample preparation and segments generation**

After marking unique ligatures on paper, a separate document of these ligatures is made. 20 copies of this document are printed and scanned to generate the train data. The font size for this data is 36 points and scanned on 300 DPI. All pages are scanned and then ligatures are isolated from text lines. Batch segmentation is used to get segments of these training ligatures. Along with these segments their frames and DCT files are also generated to accomplish the training procedure.

### **6.2.4. Clustering**

This is very important module in training process. Segments of training data have been generated and now they need to be grouped for model training. Ligatures are broken into many segments and it becomes very time consuming process to manually group them. Hence, it was decided to use some clustering mechanism to ease trainer.

#### **6.2.4.1. K-Means**

For experimentation, K-means clustering technique has been used for grouping of segments. Segments of initial 6 classes were clustered to check the technique. The method helped a lot but there were some problems faced during the process. First of all the clustering technique requires number of clusters in advance. Most of the time number of unique clusters is unknown before grouping. A random guess can be given to start the

algorithm but it required a lot of manual working. Also simple distance measures didn't give very good results.

The other major problem faced during this work was huge memory and processing requirements. In order to check every segment's distance from others, large tables are required to be calculated. All this data needs to be maintained and stored in memory. During experimentation it was found that more than 2000 segments take almost an hour to be grouped and sometime program is stuck near completion. One possible solution was to perform clustering in multiple sessions but the tracking of the procedure becomes very tedious and eventually takes similar amount of effort as taken in manual clustering.

#### **6.2.4.2. HTK based training with manual pruning**

After looking at the effort and process of K-means it was decided to use some basic heuristics along with automatic grouping of segments using HTK. Although it is not fully automatic training but it provides reasonable help for trainer.

The basic difference between previous research and this study is thickness of segments. We decided to use the benefit of thick stroke. All segments produced by the training data are divided with respect to their dimensions. Maximum and minimum values of height and width are found in train data. 4 different groups are formed using paired values of height and width i.e. (MaxH, MaxW), (MaxH, MinW), (MinH, MinW), (MinH, MaxW). Segments having nearest values to any of these paired values are sent to their respective group. When this algorithm is completed for one pass, another pass for making subgroups is also applied on data. Usually two passes are enough to make some reasonable groups of data. After this, manual pruning is done to cater boundary cases.

Once there is some logical grouping of segments is done the problem has been divided. Now segments in each group are ordered with trained model of HTK. All the segments are tested using recognition engine and grouped according to result. There are many segments which were not trained in the system but they are misclassified to same segment most of the time. Hence, all such segments are grouped together. The technique also helps in identifying those segments which have already been added to recognition engine. Eventually the last manual pruning is performed to isolate old and new segments. Thick segments are very clear in their shape as compared to thin ones used in previous research. So, manual passes are quickly done using "thumbnail view" of image.

## 6.2.5. Model training

Recognition model training is next step after getting clustered segments of one class. As mentioned earlier that frames and DCT files have been generated during batch segmentation process, now these files are also grouped according to their clusters. Data present in each cluster is used to construct one HMM. All trained HMMs contribute to develop model of recognition. After completion of first 6 classes total 48 HMMs were made. At the end of training, we have total **228** segments covering **21** classes. A complete model is developed for recognition of these segments.

Diacritics have been handled separately. 20 samples of each diacritic are collected from different segments and then a separate model is trained for their identification. The separate engine ensures very high accuracy of recognition of these marks. Total 7 diacritical marks are trained in this 2<sup>nd</sup> HMM model.

HTK provides interface for HMM training but for every segment an empty HMM model is required to be constructed with dummy value and states. After that values are updated on the bases of sample data. The training process is very well explained in previous research [1, 22]. The trained model was expected to be very large and it was recommended by previous researchers to use semi automatic training method. So an application was developed to facilitate the training process and minimize manual work as much as possible. Following figure shows the interface of training application.

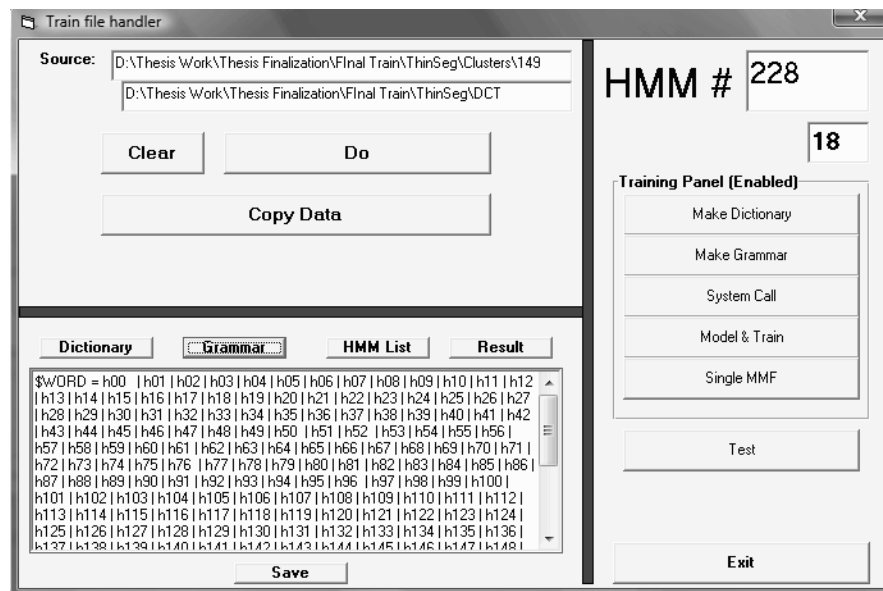


Figure 30: Training application

### 6.3. Testing data

Testing of the system has been carried out again on populated data as 36 point size is not very common in printed material. The goal of this research was to prepare a system which can reuse segments as building blocks and recognize untrained ligatures as well. In addition to that a reasonable cover for the frequent words was also required. The fulfillment of this goal is tested by selecting most frequent words of Urdu corpus. The highest frequency word list of Urdu corpus is released by CRULP. It was used as benchmark of testing.

All unique ligatures from these words are extracted and then printed for testing. There are 2494 ligatures in total out of which 1861 were used in training and 633 are new for the system. These test samples are printed from different printer and scanned with another scanner not used in training process. So the test data is completely different as far as samples are concerned. The 1861 ligatures which were used in training are called seen data and 633 other ligatures are categorized as unseen data.

### 6.4. Testing application

Although all of the modules were developed separately in C++ but for the testing purpose an interface application was developed to facilitate the testing procedure. The interface provides the intermediate outputs of each module to identify the error if occurred. Testing is carried out on individual ligature sample to simplify result generation. Due to very large model and frequent I/O operations used in this application, result is produced slowly. But this is only a testing application and by using some good engineering techniques the speed of application can be improved. Application is shown in the following figure.

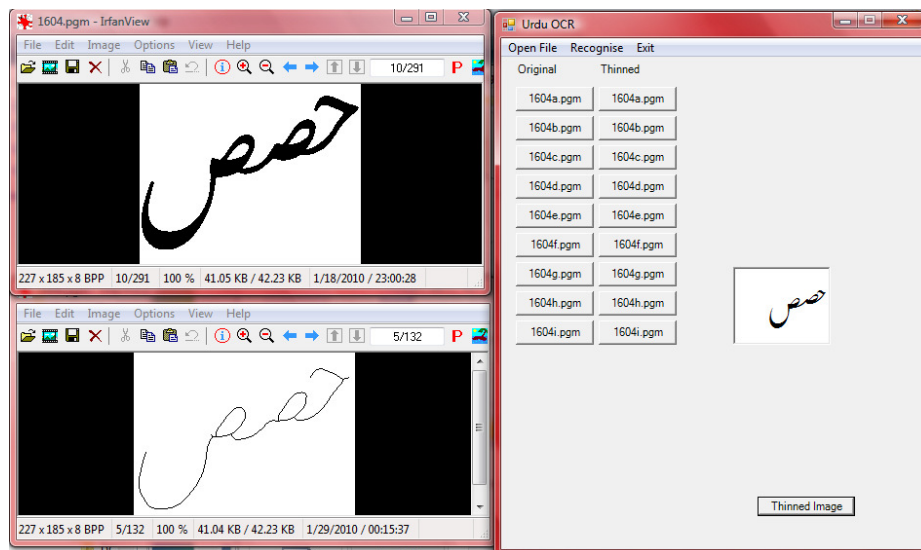


Figure 31: Testing application

## 6.5. Test plan

There have been three different modules of our application which can cause errors in recognition. First of all segmentation module can generate unseen segments. After that recognition engine can also return false ID of segment due to confusion with other segments. Finally the post processing unit which includes ligature sorting and diacritic association can produce wrong sequence of characters. Previous research carried out only segmentation and recognition process. Character identification is being done first time in this study. In order to evaluate the results seen and unseen ligatures are tested for these three modules. Test results are given in next section.

\*\*\*\*\*

## Chapter 7. Results and Error Analysis

In this section first we will discuss the reported results and major causes of errors with their solutions. Seen and unseen ligatures are tested for all three modules. During the whole process, if one ligature is segmented wrongly it is marked as **segmentation fault** and is not dispatched to recognition module. Similarly if any one segment of a ligature is not identified correctly by recognition engine then it is marked as **recognition error**. Finally the ligatures passing the post processing module are the correct identified ligatures. The failed ligatures are marked as **post processing error**. These errors are listed separately for seen and unseen samples and also collectively for one module.

Along with error reporting of each module, frequent issues have also been discussed. Possible causes of failure are elaborated and doable solutions are suggested to improve the performance of our system. In recognition section, a comparative analysis with older results has also been carried out.

### 7.1. Segmentation

All of the training ligatures have been processed by segmentation module. There is not any crash for thinning so complete set of 2494 ligatures came as input. Segmentation fault has been experienced in 48 ligatures out of 1861 seen ligatures. While only 12 out of 633 ligatures from unseen ligatures are not segmented correctly. The error analysis carried out after segmentation module is summarized in Table 3.

**Table 3: Segmentation results**

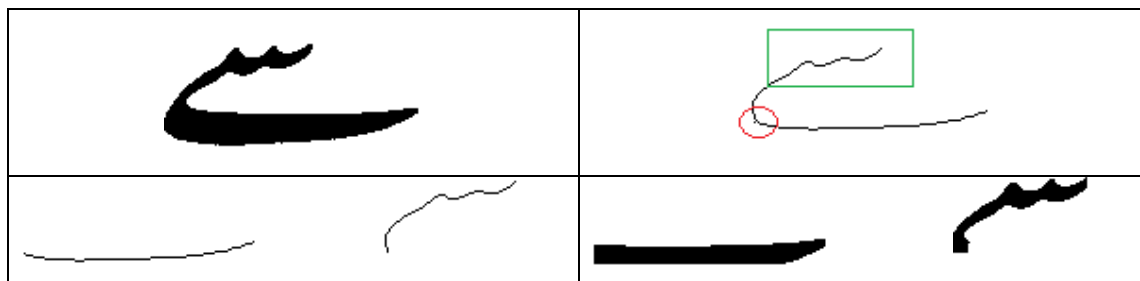
	<b>Seen</b>	<b>Unseen</b>	<b>Total</b>
Sample Tested	1861	633	2494
Segmentation Fault	48	12	60
Error	2.57%	1.83%	2.40%
Accuracy	97.42%	98.10%	97.59%

#### 7.1.1. Issues

The criterion for segmentation failure was very simple. If a ligature is segmented into already known segments (on which the system is trained) then the ligature segmentation is marked correct. The segmentation technique is very simple but it is very vulnerable. A slight change in the ligature may cause change in the thinned stroke and hence, different segments are generated. During training of system, various possible segments were generated. Most of

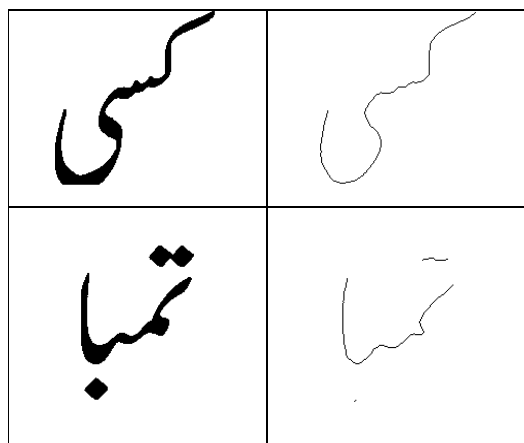


those possible segments are trained but test data may have some noise or distortion to generate some new segments e.g. the following figure is an example of strange segmentation.



**Figure 32: A wrong junction point was made during the thinning process**

The figure above shows how segmentation can be failed because of slight change in the shape. The character “س” was supposed to be segmented but it was flattened. Although this variation was exposed during the training process and is captured but the purpose of showing this example was to give an idea how segmentation can suffer because of small changes in ligature’s shape. Now we will share some under segmentation failure examples which occurred during testing process.



**Figure 33: Segmentation error because of absence of junction point**

These kinds of segmentation error can occur due to absence of junction points. The thinning algorithm works very well here in finding the center stroke of ligatures. But the problem arises when all the corners are rounded off and whole ligature is returned as the only segment.

During the training process, some other issues were also faced. Sometimes out of 20 samples only one or two sample are not segmented well. But it becomes hard to cover all

possibilities, because if the whole ligature had to be trained then there is no need to segment at the first place. Same is the case with over segmentation case. The main cause of change in thinned stroke is noise but there are more reasons for this kind of behavior. The ligatures of Noori Nastaleeq are written with hand and stored in computer. So although we have trained one ligature “ﻻ” but it becomes hard to get the same segment of similar ligature “ﻻ” whose characters are belonged to same classes. The inconsistency of strokes also makes it difficult to cope with more complex ligature formations.

### **7.1.2. Suggestions**

One possible solution to this problem is selection of more robust segmentation technique. The current segmentation has served the purpose very well. The test results of initial research showed promising results. But with the addition of more classes the number of possible segments kept increasing. And eventually it affected the whole process of recognition. The recognition model has become very large and it took a lot of time to be trained. In addition to that, post processing unit had to capture all possible combinations along with diacritics.

Under segmentation and over segmentation has also been observed in this technique. This process can be improved with consultation of some calligrapher. One possibility is to identify the key features of joints in Urdu writing and cut the characters instead of arbitrary junction point cutting. But again the inconsistent behavior of hand written font can be a hurdle to correctly identify segmentation point.

Another possibility is to use diacritics for identification of character boundaries. These marks can be very helpful in determining the correct classes. For example one dot above any cup like structure ensures there is “ن” present inside the ligature. Similarly other diacritics can be good heuristic to identify probable segmentation points. A good segmenter can not only increase segmentation accuracy but also it will make recognition and post processing relatively easy.

## **7.2. Recognition**

Ligatures failed in segmentation module are eliminated from recognition testing. Total 1813 seen and 621 unseen ligatures are taken as input in this module. Recognition accuracy of seen

data is better this time as anticipated. The observation carried out after recognition module is summarized in the following table.

**Table 4: Recognition results**

	Seen	Unseen	Total
Sample Tested	1813	621	2434
Recognition Fault	132	58	190
Error	7.28%	9.33%	8.62%
Accuracy	92.71%	90.66%	92.19%

### 7.2.1. Issues

The recognition results for seen and unseen data didn't show equal accuracy. The recognition model has been trained with numerous samples and data is collected from valid ligatures. Also, with the addition of new classes, recognition of already trained segments has been constantly checked. In case of similarity, multiple segments were merged with each other to minimize the confusion. But unseen segments do vary in shapes because every ligature is stored separately in Noori Nastaleeq. Ligatures are not constructed with any algorithm. These are hand written, pre stored glyphs used to write Urdu words. Hence, unseen data is more error prone.

The errors of recognition system are generally because of some less trained sample size for particular segments. Another error is observed when a longer segment has an identical initial frame sequence with some shorter segment; the longer is misclassified as shorter one. Figure 34 is an example of such case.



**Figure 34: The segment in the left image is misclassified with the right one**

The image on the right requires one more segment to construct the whole character, so this misclassification leads to a recognition fault.

Recognition errors have been observed in some particular classes. The recognition of class “س” and “م” has been relatively poorer than other classes. The class “م” has caused errors in segmentation as well in recognition. Especially its initial position is harder to identify for the recognition engine. The shape is so small itself that it can be easily classified with some other segment.

## 7.2.2. Class-wise comparative analysis

The recognition results are better than the results reported by previous researchers with system of 6 classes. With addition of 15 more classes and 168 more segments the recognition is still above 92% for seen data. It has been possible because of thick stroke segments and various samples for each segment.

In order to compare these recognition results with previously reported ones, we have extracted results of some classes as well. In following table recognition rate of previous 6 and 3 new classes are elaborated. We selected these 3 classes for discussion because of their relatively lower accuracy.

**Table 5: Class-wise recognition results**

No.	Class	Occurrences	Recognized	Accuracy %	Previous results
1.	ا	419	419	100	99.5
2.	ب	1163	1144	<b><u>98.36</u></b>	<b><u>94.9</u></b>
3.	د	146	146	100	99.3
4.	ص	144	141	<b><u>97.9</u></b>	<b><u>80</u></b>
5.	ع	201	196	97.5	94.7
6.	ی	1068	1046	<b><u>97.9</u></b>	<b><u>88.6</u></b>
7.	م	585	553	94.5	NA
8.	ح	187	172	92.9	NA
9.	س	593	531	89.54	NA

The comparison shows that accuracy rates are climbed significantly for some classes. It has been possible because of using thick and original segments. The number of black pixels is increased by very high ratio. One possible approach for this is to compare the frame sizes. 64 data values per frame were used to capture thin segments. We have 225 data values in a frame to accommodate original stroke. In addition to that overlapped framing is opted which ensures more coherence in data points. The frame sequence gives the data sequence for recognition engine. Hence, when two frames are overlapped and share some data the sequence becomes very unique leaving very less chances of confusion. Improvement in recognition of class “ص” is because of capturing four different

shapes. Although the shapes look similar when thinned but thick stroke shows some change in size and angle. Same is the case with “ﻉ”. The confusions reported in previous research are because of slight change in thinned stroke which leads to misclassification. But thick strokes for confused segments are very much different and hence, separately identified. In next subsection some suggestions are given to improve recognition for classes which showed more errors.

### **7.2.3. Suggestions**

In order to increase recognition accuracy, one possibility is to use some image processing techniques. Some shape heuristics can be added along with recognition engine to enhance the recognition results. For example masking can help in reduction of errors in recognition of “ﻡ” class. Also class “ﺏ” has similar shapes in most of the ligatures, but due to different segmentation points it is segmented differently. A shape detection mask can give some hints about the presence of it in ligature. But the integration of these image processing techniques with post processing engine would be a challenge.

The main cause of misclassification is very large model trained for 228 segments. This model can be divided into sub models by setting some distinguishing features. For example dimensions of segments can be one feature to consider. Also if diacritics are associated in preprocessing phase with some high accuracy, then dots bearing segments can be sent to separate recognition engines. It will help in simplifying the model and eventually reduction of misclassification.

As discussed earlier a more robust segmentation technique can also decrease the redundant segments. The problem of misclassification of longer segment to shorter one can be reduced with accurate segmentation technique.

## **7.3. Post processing errors (Ligature level)**

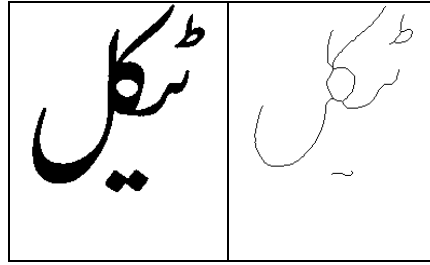
After ruling out all ligatures having recognition fault, we had 1681 seen and 563 unseen test samples. All of segments in these ligatures have been correctly identified. The character level recognition is carried out in this module with diacritic association. The observation carried out after post processing module is summarized in the Table 6.

**Table 6: Post processing results**

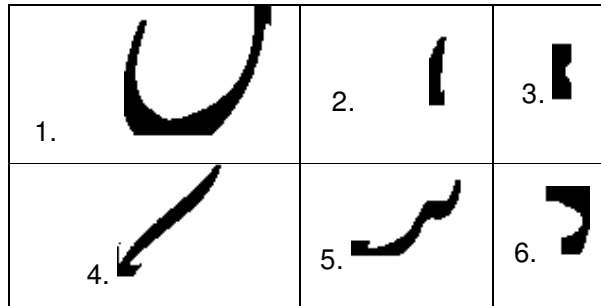
	Seen	Unseen	Total
Sample Tested	1681	563	2244
Post processing Fault	198	94	292
Error	11.77%	16.69%	13.01%
Accuracy	88.23%	83.31%	86.98%

### 7.3.1. Issues

Post processing module has been suffered with two types of errors. First one is sequence generation of segments and other is diacritic association. Segments are sorted with respect to spatial positioning but the algorithm fails in some cases. When the order of segments is not right a segment of one character comes between two segments of some other character and Unicode acquisition is failed. One example of this scenario is given below in Figures 35 and 36.



**Figure 35: Ligature with its thinned image**









**Figure 36: Sorted segments of ligature**

In the given example segment 1,2,3,4 are arranged in correct order. 1 and 2 combine to make “ل” but 3, 4 and 6 are required to make “ك” which is not generated. Segment 5 is present below 6 and also positioned on the left of 6 that is why it comes earlier in sequence. Hence, “ك” is misclassified. This is because of little difference in the shape of this particular ligature.

Other character combinations with “ك” do not behave similar all the times.

Other post processing problem is flaws in diacritic association. The diacritic association is done with horizontal overlapping and direct distance measuring from center of mark to center of segment. Sometimes there are two segments overlapping the same diacritic but the closer segment is not the original one to bear that mark. Majority of such cases are with “ے”, “ش” and sometimes with long handle of “ک”. Examples of such cases are discussed below.

		
1.	2.	3.
		
4.	5.	6.

**Figure 37: Examples of diacritic association error**

We will discuss the problems in given examples one by one. In example 1, highlighted dot is covered with two segments. One is “ج” and other is “ے”. The association algorithm checks the distance and dot is associated with “ے” and it becomes “بے” which is a valid possible combination. It results in wrong ligature construction. Example 5 is another instance of linking the dot with wrong segment. The horizontal coverage is good but the distance rule again failed the correct attachment.

In example 2, the diacritic mark is shifted from the segment without the presence of any other mark. Diacritic shifting is managed with the presence of other dots. But in this case it cannot be associated with correct segment. Eventually it is resulted into an error. Example 4 and 6 are also cases of extended shifting. In image 6, dot is not associated with its segment and the last segment is identified as “ل” and dot is associated with other segment resulting in meaningless ligature.

The dots encircled in example 3 cannot be associated with any segment because of over segmentation of “ش” and all three dots were not associated with one segment. Instead they were attached with two different segments resulting “ت” and “ن”. This example was very rare in whole test data but was an error in ligature construction.

### **7.3.2. Suggestions**

Diacritic algorithm designed for this study failed for different scenarios but it was very first exercise. The algorithm can be improved with detailed study of Noori fonts. Interestingly these problems are not very common with other Nastaleeq fonts. As all ligatures are written and stored so it is hard to devise some common rules. A lot of exceptions are required to cover all individual cases.

The sequence problem can be solved by indentifying all wrong sequence generations of that type and make an exception list to correct the order. It is hard to find some other heuristic for correction of order.

### **7.4. Summary**

We have seen that segmentation module failed to pass 60 samples out of which 48 are seen and 12 new ligatures. The percentage accuracy is higher for unseen test data. But recognition results are better for seen data than unknown ligatures. The factor behind these kinds of results is understandable. A lot of segmentation errors have been identified in longer ligatures. Unseen ligatures are mostly from 1-4 character combinations. Bigger ligatures are common in training and testing data. Same is the case with recognition problems. Some particular classes are difficult to be recognized by the recognition. A huge model for recognition engine makes it harder to identify the exact ID of a segment with similarities.

\*\*\*\*\*



# Chapter 8.Future Work and Conclusion

## 8.1. Future directions

A lot of work can be done for the advancement of this research study. A few of the recommendations are given below.

1. A preprocessor engine can be made to aid the recognition method, with the help of some advance technique of image processing. That can increase recognition accuracy.
2. Size independent module can be added to enable this application for a variety of font sizes.
3. An intelligent post processing unit with robust diacritic association algorithm can increase the accuracy of the system.
4. Multiple recognition engines can be trained for higher accuracy and quick results. Classification of characters can be done with the help of diacritic association prior to recognition.
5. Noise removal and skew correction can be added. It will help to process real data.
6. Recent research has been carried out using Tesseract OCR engine, which is easy to train and considered more accurate. It is useful to adopt a predesigned recognition engine instead of designing a complete one again.

## 8.2. Conclusion

This study was next step for segmentation based OCR for Noori Nastaleeq. The work done has been a continuation of already proposed techniques with refinements in different modules. Segmentation technique is explored thoroughly in this study and its pros and cons have been revealed. Optical Character Recognition is an open and challenging field for researchers to implement new ideas. The basic challenges for Urdu OCR are segmentation and handling the diacritical marks. Both of these challenges are font dependent. Hence, a detailed font study can help in finding good solutions for these challenges.

\*\*\*\*\*

## Chapter 9. References























- [1]. Javed, S., Maqbool, A., Jameel, S. and Qureshi, S., A. “*Urdu Nastaleeq OCR*”, BS final year report, 2005
- [2] “Urdu, A language of Pakistan” available at [http://www.ethnologue.com/show\\_language.asp?code=urd](http://www.ethnologue.com/show_language.asp?code=urd) [Visited on 6<sup>th</sup> May,2008]
- [3] “Urdu (اردو)” available at <http://www.omniglot.com/writing/urdu.htm> [visited on 6th May, 2008]
- [4] “Urdu and Languages of Pakistan” available at <http://tlt.its.psu.edu/suggestions/international/bylanguage/urdu.html#script>
- [5] Hussain, S. Complexity of Asian Writing Systems: A Case Study of Nafees Nasta’leeq for Urdu
- [6] Wali, A. and Rehman, S. “*Implementation of Reverse Chaining Mechanism in Pango for Rendering Nastaliq Script*”, in Second Workshop of Computational Approaches to Arabic Script-based Languages, Stanford University, USA, 2007.
- [7] Hussain, S. and Afzal, M. “*Urdu Computing Standards: UZT 1.01*”, in the Proceedings of the IEEE International Multi-Topic Conference, Lahore, Pakistan, 2001.
- [8] Wali, A. and Hussain, S. “*Context Sensitive Shape-Substitution in Nastaliq Writing system: Analysis and Formulation*”, in the Proceedings of International Joint Conferences on Computer, Information, and Systems Sciences, and Engineering (CISSE), 2006.
- [9] Localization Software resources, available at <http://www.crupl.org/software/localization.htm> [visited on 8th September, 2008]
- [10] Sauvola, J. and Pietikainen, M., “*Adaptive Document Image Binarization*”, Pattern Recognition 33(2), 2000
- [11] Nasta`liq typesetting <http://www.nationmaster.com/encyclopedia/Nastaleeq>
- [12] Durrani, N. “*Typology of Word and Automatic Word Segmentation in Urdu Text Corpus*” unpublished MS thesis report, 2007.
- [13] Shafait, F., Beusekom, J., Keysers, D. and T. M. Breuel, “*Page Frame Detection for Marginal Noise Removal from Scanned Documents*”, in 15th Scandinavian Conference on Image Analysis, Aalborg, Denmark, 2007.
- [14] AIM report on OCR, published in 2000 “Optical Character Recognition”
- [15] Amin, A., Fischer, S., Parkinson, S. and Shiu,R. “*Fast algorithm for skew detection*”
- [16] Husain, S.A. and Amin, S.H., “*A Multi-tier Holistic Approach for Urdu Nastaliq Recognition*”, IEEE INMIC, Karachi, Pakistan, 2002.
- [17] Shah, Z. and Saleem, F. “*Ligature Based Optical Character Recognition of Urdu, Nastaleeq Font*”, in the Proceedings of International Multi Topic Conference, Karachi, Pakistan, 2002.















- [18] Pal, U. and Sarkar, A. "*Recognition of Printed Urdu Text*", in the Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR), 2003
- [19] Jörg Schulenburg, GOCR-documentation, 2002.
- [20] Beurel T. "*The OCRopus OpenSource OCR System*" IUPR Publication Repository, 2009.
- [21] Breuel, T. M. "*Robust Least Square Baseline Finding using a Branch and Bound Algorithm*", in Document Recognition and Retrieval VIII, SPIE, San Jose, 2002.
- [22] Javed, S., "*Investigation into a Segmentation Based OCR for the Nastaleeq Writing System*", unpublished MS thesis report, 2007.
- [23] Bokser, M., "*Omnidocument Technologies*", Proceeding of IEEE 80(7), IEEE, USA, Jul 1992.
- [24] Khorsheed, M. and Cocksins, W. "*Spectral Features for Arabic Word Recognition*" in the proceedings of the International Conference on Pattern Recognition, 2000.
- [25] Elgmmal, A., and Ismail, M. "*A Graph-Based Segmentation and Feature Extraction Framework for Arabic Text Recognition*" in proceeding of IEEE conference on Document Analysis and Recognition, 2001
- [26] Alma'adeed, S., Higgins, C. and Elliman, D. "*Recognition of Off-Line Handwritten Arabic Words Using Hidden Markov Model Approach*" Proceedings of the 16 International Conference on Pattern Recognition (ICPR'02), 2002
- [27] Zidouri, A. "*ORAN System: A Basis for an Arabic OCR*" In the proceeding of Intelligent Multimedia, Video and Speech Processing, 2004
- [28] Rashaideh, H. "*Preprocessing phase for Arabic Word Handwriting Recognition*" Institute for Informatics and Automation, 2006.
- [29] Lopresti, D., Nagy, G., Seth, S. and Zhang, X. "*Multi-Character Field Recognition for Arabic and Chinese Handwriting*", 2006
- [30] Schambach, M., Rottland, J., and Alary, T., "*How to Convert a Latin Handwriting Recognition System to Arabic*" The 11<sup>th</sup> International conference on Frontiers of handwriting recognition, 2008
- [31] Official website of OCRAD <http://linux.die.net/man/1/ocrad> [visited on 7th July, 2009]
- [32] Shamsheer, I., Ahmad, Z., Orakzai, J.K. and Adnan, A. "*OCR for Printed Urdu Script Using Feed Forward Neural Network*" In the Proceeding of worlds academy of Science, Engineering and Technology, 2007
- [33] Husain, S., A., Sajjad, A. and Anwer, F. "*Online Urdu Character Recognition System*" IAPR conference on Machine Vision Applications", 2007
- [34] Javed, S., Hussain, S. "*Improving Nastaleeq-Specific Pre-Recognition Process for Urdu OCR*" In the Proceedings of 13th IEEE International Multi topic Conference 2009 (INMIC 2009), Islamabad, Pakistan, 2009
- [35] Jang, B.K., Chin, R.T. "One-Pass Parallel Thinning: Analysis, Properties, and Quantitative Evaluation" IEEE Trans. Patt. Anal. Machine Intell. , Vol. 14, No.11, pp.869-885, 1992.
- [36] Inpage documentation <http://www.inpage.com/faq.htm> [visited on 20th December, 2009]

[37] Official website of Tesseract <http://code.google.com/p/tesseract-ocr/> [visited on 20th December, 2009]

\*\*\*\*\*

**Annexure**  
**HMM Identification Numbers**

ID	Shape
0.	
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	
11.	
12.	
13.	
14.	
15.	
16.	
17.	
18.	
19.	
20.	
21.	

22.	
23.	
24.	
25.	
26.	
27.	
28.	
29.	
30.	
31.	
32.	
33.	
34.	
35.	
36.	

37.	ی
38.	نی
39.	ی
40.	ا
41.	س
42.	ا
43.	سا
44.	سا
45.	سی
46.	ر
47.	ہ
48.	ر
49.	حی
50.	حی
51.	سی

52.	سی
53.	ہ
54.	ر
55.	ر
56.	ر
57.	ح
58.	حا
59.	لا
60.	ن
61.	و
62.	ر
63.	ر
64.	ی
65.	ر
66.	ر
67.	ی
68.	م
69.	ر
70.	ر

71.	
72.	
73.	
74.	
75.	
76.	
77.	
78.	
79.	
80.	
81.	
82.	
83.	
84.	
85.	
86.	
87.	
88.	
89.	
90.	
91.	
92.	

93.	
94.	
95.	
96.	
97.	
98.	
99.	
100.	
101.	
102.	
103.	
104.	
105.	
106.	
107.	
108.	
109.	
110.	
111.	
112.	
113.	
114.	
115.	
116.	

117	
118	
119	
120	
121	
122	
123	
124	
125	
126	
127	
128	
129	
130	
131	
132	

133	
134	
135	
136	
137	
138	
139	
140	
141	
142	
143	
144	
145	
146	
147	



148	ک
149	ک
150	ک
151	ک
152	ک
153	ک
154	ک
155	ک
156	ک
157	ک
158	ک
159	ک
160	ک
161	ک
162	ک
163	ک

164	ک
165	ک
166	ک
167	ک
168	ک
169	ک
170	ک
171	ک
172	ک
173	ک
174	ک
175	ک
176	ک
177	ک
178	ک
179	ک

180	و
181	و
182	و
183	و
184	و
185	و
186	و
187	و
188	و
189	و
190	و
191	و
192	و
193	و
194	و
195	و

196	و
197	و
198	و
199	و
200	و
201	و
202	و
203	و
204	و
205	و
206	و
207	و
208	و
209	و

210	ل
211	و
212	ک
213	می
214	ک
215	م
216	کی
217	لہ
218	سما
219	محا
220	س
221	س
222	س
223	س
224	ک
225	م

226	ر
227	ہ
228	و
Diacritics Number	
D0.	◆
D1.	◆◆
D2.	ب
D3.	ۛ
D4.	ۛ
D5.	ۛ