Abstract—The electronically available Urdu data is in image form which is very difficult to process. Printed Urdu data is the root cause of problem. So for the rapid progress of Urdu language we need an OCR systems, which can help us to make Urdu data available for the common person. Research has been carried out for years to automate Arabic and Urdu script. But the biggest hurdle in the development of Urdu OCR is the challenge to recognize Nastalique Script which is taken as standard for writing Urdu language. Nastalique script is written diagonally with no fixed baseline which makes the script somewhat complex. Overlap is present not only in characters but in the ligatures as well. This paper proposes a method which allows successful recognition of Nastalique Script.

Keywords—HMM, Image processing, Optical Character Recognition, Urdu OCR.

I. INTRODUCTION

The text within the image is non-editable. Due to this non-editable nature of text, information retrieval, collection and sharing become a big issue. So we need a mechanism to convert this text image into machine editable text form. OCR (Optical character recognition) helps to convert the text image such as scanned document, electronic fax files, and picture of document taken from cameras into text file that can be opened in any word processor or text editor.

The process of OCR can be divided into three main processes [1]
1. Preprocessing
2. Recognition
3. Post Processing

Different preprocessing techniques are applied to extract noise and distortion free image. The text areas are extracted and fed to different classifiers for recognition. The result of classifiers is significantly improved in the last step of post processing.

II. URDU WRITING SYSTEM

Urdu is the national language and lingua franca of Pakistan. Urdu is bidirectional as words are written right to left where numbers are written left to right as shown below [2].

The extended Arabic character set of Urdu join together to form words/ligatures ([3], [4]). Urdu is very context sensitive, that is, characters change their shapes depending upon the context in which it is written. Due to the context sensitive nature, the shape of character may be categorized as isolated, initial, medial and final. Apart from the dots of the character, diacritical marks are used for the proper pronunciation of Urdu word [5]. The diacritics may appear above or below a character.

Following table shows the different unique classes.
Urdu can be written with different styles. Nastalique, Naskh, Kofi, sals, devani and Raka are the different styles of writing Urdu [6]. Today most commonly used script are Nastalique and Naskh. The figure below tells how these different styles look like.

A. Nastalique Script

Nastalique is a combination of two different fonts, Naskh and Taleeq. It was initially created by Mir Ali Tabrezi. The calligrapher tends to increase the beauty of the script and making it a complex script. It is highly cursive and context sensitive in nature. Some of the characteristics are as follows:

1. It is written diagonally from top right to bottom left. That means all the ligatures are tilted at a certain angle towards the right side as shown below. Due to this diagonal nature the nastalique consumes less horizontal space as compared to Nask [2].

2. The joins are formed by cusp-like shapes, which are concave upwards and have their initial end higher than the final end [2].

3. Overlapping problem is present in characters and ligatures [2]. For example kaf of the word “kisam” is overlapping “meem” of tamam.

4. Context Sensitive nature of nastalique compels the letters to adapt different shapes [6]. Following figures are taken from [2]
III. METHODOLOGY

Mostly the work that is available today is on Arabic OCR. Little effort has been made on Urdu OCR and especially on Nastalique Urdu OCR [17, 18, 19, 20, 21, and 22]. The cursive and context sensitive nature of the script complicates the situation even further.

For recognizing cursive script, there are two main approaches to deal with connected characters in a word.

1. Segmentation based Approach
2. Segmentation free Approach

Due to highly cursive nature the segmentation of Urdu script is not a trivial task. So we will discuss segmentation free approach in which the ligature as a whole is used instead of segmenting it into smaller units. Different pattern matching techniques are used to classify the pattern. Most commonly used method in this approach is to extract features from the image and feed them to some recognizer for identification purpose. Three types of features are normally extracted which are transformational [7, 8], statistical [9, 10, 15] and structural [13].

In our research, we extracted global transformational features from a non segmented ligature and then fed them to Hidden Markov Model (HMM) recognizer. HMM has an ability to perform recognition with great ease and efficiency. Language-independent training and recognition methodology was a major reason of using HMM in developing recognition technology [11]. Hidden Markov model Tool Kit (HTK) [12] is used to implement HMM.

The Process of HMM recognition is divided into two tasks.

1. Training
2. Recognition

A. Training

Before starting the recognition process, the system is properly trained on each unique HMM (training data) [16]. Each connected body is considered as a separate HMM. Connected body may be a diacritic or a main body. A separate recognizer for diacritics and main bodies are built. Diacritic Recognizer is trained on one dot, two dots, three dots, Shad, Mad, Secondary Stroke of hey, Secondary Stroke of gaf; Khari Zabar, Do Zabar, Hamza and small toey. Main Body Recognizer is trained on 1500 high frequency ligatures which vary from one character ligature to seven characters ligature. In order to cater all sorts of noise and variations in the image we need to collect at least 30 samples of each HMM model.

The figure below gives the overall flow of the process.

For main body Recognizer not only the HMM models are trained but a lexicon is also maintained at the time of training which tells about all the character classes within each model. Entry for each HMM model is incorporated in the lexicon. The following table gives some of the entries of lexicon to identify the character classes with in the ligature.
Once all the models are properly trained, the system is ready for recognition. The overall flow of recognition process is given in the figure below. After separating and re-associating the diacritics and main body, each connected body is further divided into small chunks/windows called frames. The discrete cosine Transform (DCT) is calculated for each frame and then fed to HMM recognizer. The information regarding the location of the diacritic is also stored which will be in the recognition phase.

Diacritics are divided into vertical windows of size 9x25 whereas main body is divided into vertical windows of size 8x90. 1 bits are padded to complete the window. The direction of frame traversal is from right to left as shown below. Let’s take an example of the Urdu letter “ﷺ” and see how framing is done for its diacritic and main body.

**C. Feature Matrix**

Once the diacritics are recognized, feature vector is prepared according to the information of diacritics. If no diacritic is present then the vector is ‘0’. If the diacritics are present then after recognizing them we make the vector as shown below.

We used the following codes for making feature matrix:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>1 dot above</td>
</tr>
<tr>
<td>02</td>
<td>2 dots above</td>
</tr>
<tr>
<td>03</td>
<td>3 dots above</td>
</tr>
<tr>
<td>04</td>
<td>1 dot below</td>
</tr>
<tr>
<td>05</td>
<td>2 dots below</td>
</tr>
<tr>
<td>06</td>
<td>3 dots below</td>
</tr>
<tr>
<td>07</td>
<td>Shad</td>
</tr>
<tr>
<td>08</td>
<td>Mad</td>
</tr>
<tr>
<td>09</td>
<td>Secondary Stroke of Hey</td>
</tr>
<tr>
<td>10</td>
<td>Secondary Stroke of Gaaf</td>
</tr>
<tr>
<td>11</td>
<td>Khari zabar</td>
</tr>
<tr>
<td>12</td>
<td>Do zabar</td>
</tr>
<tr>
<td>13</td>
<td>Hamza</td>
</tr>
<tr>
<td>14</td>
<td>Small toey</td>
</tr>
<tr>
<td>15</td>
<td>1 dot in the middle</td>
</tr>
<tr>
<td>16</td>
<td>2 dots in the middle</td>
</tr>
<tr>
<td>17</td>
<td>3 dots in the middle</td>
</tr>
<tr>
<td>18</td>
<td>3 dots above in the middle</td>
</tr>
<tr>
<td>19</td>
<td>3 dots below in the middle</td>
</tr>
<tr>
<td>20</td>
<td>Hay in the middle</td>
</tr>
<tr>
<td>21</td>
<td>Hamza in the middle</td>
</tr>
<tr>
<td>22</td>
<td>Small Toay in the middle</td>
</tr>
</tbody>
</table>

---

World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:4, No:10, 2010

1517

ISNI:0000000091950263
For example the feature matrix for the ligature “غتبٹ” is give as below.

Feature Matrix:

<table>
<thead>
<tr>
<th>04</th>
<th>05</th>
<th>14</th>
<th>01</th>
</tr>
</thead>
</table>

D. Recognizing the ligature

After recognizing the main body, the already developed lexicon is used to identify the classes of the characters with in the ligature. Once the classes are identified, next step is to exact characters within the ligature. For this purpose rule based recognizer is used. Feature matrix along with the identified classes is fed to rule based recognizer which then correctly recognizes the ligature. Some of the rules are listed below.

<table>
<thead>
<tr>
<th>Sr.</th>
<th>Identified classes</th>
<th>Feature Matrix</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(0) class+ (ـ) class</td>
<td>01</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>(0) class+ (ـ) class</td>
<td>06</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>(و) class+ (ـ) class+(ا) class+ (ـ) class</td>
<td>04 05 14 01</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>(و) class+(ـ) class+(ا) class+(ـ) class</td>
<td>04 05 14 02</td>
<td></td>
</tr>
</tbody>
</table>

As the characters are identified, their respective Unicode is also written in the file which gives us the required ligature.

Let’s take an example of the following ligature and see how it is recognized.

Fig. 12  Ligature to be recognized

First of all diacritics and main bodies will be separated and location of both the diacritics is store. The first single dot is marked as middle and two dots as above. Then main body and all the diacritics are sent to their respective recognizer.

Main body recognizer will give us h061 as the identified model; similarly output of diacritic recognizer will be h01 and h02. These HMM numbers tell us the type of the pattern recognized. h01 and h02 tell that diacritic (1) is one dot and diacritic (2) is two dots respectively. Model h061 of main body tells that it is six characters word with the following classes.

(و) class+(ـ) class+(ا) class+(ج) class+ (ـ) class+(ـ) class

Next step is to prepare a feature vector. It has one dot in the middle and two dots above so the feature vector will look like the following.

Feature Matrix:

| 16 | 02 |

Then the Unicode for all these characters are written in the text file such that a ligature is formed.

IV. RESULT

A total of 1282 unique ligatures are extracted from the 5000 high frequency words in a corpus-based dictionary [14]. It is also confirmed that all Urdu letters are used in these ligatures in a variety of contexts. For analysis purpose three or more samples of each ligature are taken to form the text. These pages are printed in Noori Nastalique font at font size 36. The pages are then scanned at dpi 150 and then separated back into ligatures. A total of 3655 ligatures are tested and 3375 ligatures are accurately identified, giving an accuracy of 92%.

V. DISCUSSION

Testing and further analysis shows the following reasons for the errors.

Sometimes the shapes of different ligatures are so similar that they give approximately same state transition probabilities. So when these inputs are given to HMM for recognition same
state sequence of HMM is obtained. Errorneous output is thus obtained due to the slight similarity in shape. The shape of the ligatures “۱۳۰۶۱۳۰۶” are similar to each other when written in Noori Nastaleeq font. Sometimes the OCR is not able to recognize ligatures like these correctly because of the similarity in shape. Another such problem can be seen in “۲۳۳۲۳” and “۱۳۱۳۱۳”. Both these ligatures are more similar than different. Sometimes they are recognized correctly and sometimes they are not. This problem can be removed by making the sliding window over-lapping. By using overlapping windows finer details of ligature shape can also be covered.

Sometimes due to noise or poor scanning the actual shape of the diacritic is distorted. In this case, the diacritic is not recognized correctly and as a result incorrect feature vector is made. Thus the ligature is not recognized. During scanning the diacritic “۰۵۰۰” is distorted. Total number of windows and DCT values differ to great extent. This results in erroneous recognition of diacritic. Increasing the number of samples, while training, can improve the accuracy.

For recognizing a ligature we first identify its shape and then recognize it by seeing the class of feature vector which it belongs to. Sometimes it so happens that 2 different ligatures have exactly the same feature vector though position and location of dots are different. In this case the result is erroneous due to the same feature vector. In Ligatures “۲۳۲۳” and “۲۳۲۳” the shape of main body is same and the feature vector of both the ligatures is “۰۱”. Due to the same feature vector and main body, these ligatures can’t be trained and recognized at the same time. Another such problem can be seen in the ligatures “۱۳۱۳” and “۱۳۱۳”. These ligatures too have same shape of main body and the feature vector in both these ligatures is “۰۲”. Only one of them can be recognized at a time.

VI. CONCLUSION

The Nastaliq script is complex due to its context sensitive and cursive nature. The absence of baseline and complex mark placement rules makes the situation even more wors. So the extraction of Nastaliq text from the image is not a trivial task. The algorithm has been developed which successfully extracts the texts with an accuracy of 92%.

REFERENCES


World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:4, No:10, 2010