CHAPTER 8

RESULTS AND DISCUSSION

In earlier chapters we have discussed options for carrying out various sub-tasks of OCR for Gujarati script. In most of the cases we have also given results of testing the algorithms described in that chapter. Here, we collate those result to build OCR system. We have tested the end to end system on pages from books. The test results are used to propose an ideal work flow for Gujarati OCR with selection of algorithms at various stages. Text corpus generated from the scanned pages has been used to validate the output of the OCR and find the character level accuracy.

8.1 Preliminaries

Let us try to understand the edit distance which is used to compute the accuracy of OCR output.

8.1.1 Edit Distance

Given two character strings \( s_1 \) and \( s_2 \), the edit distance between them is the minimum number of edit operations required to transform \( s_1 \) into \( s_2 \). The edit operations allowed for this purpose are: (i) insert a character into a string; (ii) delete a character from a string and (iii) replace a character of a string by another character; for these operations, edit distance is sometimes known as Levenshtein distance. For example, the edit distance between “cat” and “dog” is 3.
Implementation of this concept uses the dynamic programming algorithm, where the characters in $s_1$ and $s_2$ are given in array form. The algorithm fills the entries in a matrix $m$ of the size $(|s_1| + 1) \times (|s_2| + 1)$. Where, $|s_1|$ and $|s_2|$ denotes the lengths of strings $s_1$ and $s_2$ respectively. The $(i,j)^{th}$ entry of the matrix will hold (after the algorithm is executed) the edit distance between the strings consisting of the first $i$ characters of $s_1$ and first $j$ characters of $s_2$.

The central dynamic programming step is depicted in Lines 8-10 of Figure 3.5, where the three quantities whose minimum is taken correspond to substituting a character in $s_1$, inserting a character in $s_1$, and inserting a character in $s_2$.

\begin{algorithm}
\textbf{Algorithm 8.1} To Compute Edit Distance Between Two Strings
\begin{algorithmic}
\State \textbf{Input:} Strings $s_1$ and $s_2$.
\State \textbf{Output:} Edit distance between $s_1$ and $s_2$.
\State \textbf{Process:}
\State $m[0,0] \leftarrow 0$
\For{$i = 1$ to $|s_1|$}
\State $m[i,0] \leftarrow i$
\EndFor
\For{$j = 1$ to $|s_2|$}
\State $m[0,j] \leftarrow j$
\EndFor
\For{$i = 1$ to $|s_1|$}
\For{$j = 1$ to $|s_2|$}
\If{$s_1[i] = s_2[j]$}
\State $temp \leftarrow 0$
\Else
\State $temp \leftarrow 1$
\EndIf
\State $m[i,j] \leftarrow \min\{m[i-1,j-1] + temp, m[i-1,j] + 1, m[i,j-1] + 1\}$
\EndFor
\EndFor
\State \text{return} $m[|s_1|, |s_2|]$
\end{algorithmic}
\end{algorithm}

\section{8.2 OCR Testing}

We would like to recall that this being the first effort for developing an end-to-end Gujarati OCR, we are targeting the documents with single column of text without graphics, images or any other special decorative fonts and formats. We are also assuming pages without broken and touching glyphs.

In chapter one we have described various subtasks of an OCR system. For
our experiment we have used pages from different books scanned at 300dpi resolution with 8-bit gray scale. That is total 256 gray levels. As mentioned earlier that this gray level image can have noise due to various reasons.

Mean and median filters can be used to remove such noise. As mentioned earlier mean filter may give rise to edge blurring which in turn can generate ladder effect in binarization. Hence we use edge preserving median filter to remove isolated noise pixels.

The next step is binarization to separate text from its background. We have tested ocr system by using both the binarization method presented earlier in this work and we will present the result while testing on pages from books.

Binarized image is then subjected to segmentation. As mentioned earlier that we have considered pages with single column text without any images in it. Hence we are not doing the step for text - image separation and directly proceed to the next step. As mentioned in chapter 2 we follow top down approach for segmentation. We use the algorithms mentioned earlier for line and word segmentation. It is also clear from the discussions that we need to carry out zone boundary identification and hence the next task is to do zone boundary identification.

There are different approaches described earlier for zone boundary identification. As mentioned in chapter 4 the zone boundary detection though the algorithm mentioned there has significant limitations. Hence, we are using Algorithm 3.2 for zone boundary detection for this testing.

It is clear from the statistics for classifier accuracy that fringe map features with nearest neighbor classifier gives best classification accuracy. Therefore, we have selected this combination for this testing.

Recognized glyphs are then used for text generation through the method presented in chapter 7 and unicode text with appropriate line and word breaks is given as output of the system.

Edit distance is used to measure the accuracy of the output. Unicode characters are used as bases for comparison. Text output of a page is considered a long string of Unicode characters, say $S_r$ and edit distance of this string with the manually entered and proof read text ($S_o$) for that page is computed.
Error rate $E$ in percentage, is computed as follows:

$$E = \frac{ED(Sr, S_0)}{NU} \%$$  \hspace{1cm} (8.1)

where $NU = $ No of Unicode characters in proof read text.

We have used software developed as a part of consortia mode project of The Ministry of Communications and Information Technology to measure the accuracy.

### 8.3 Results and Discussion

Following tables give results of experiments done with the work-flow mentioned above.

Table 8.1: % Error with Niblack's binarization

<table>
<thead>
<tr>
<th>Book Name</th>
<th>No of Pages</th>
<th>% Error</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prateeksha</td>
<td>76</td>
<td>4.10</td>
<td>24.8</td>
<td>14.89</td>
<td></td>
</tr>
<tr>
<td>Birbal in Bahoshi</td>
<td>254</td>
<td>7.08</td>
<td>25.5</td>
<td>13.46</td>
<td></td>
</tr>
</tbody>
</table>

Table 8.2: % Error with Sauvola and Pietaksine\'s binarization

<table>
<thead>
<tr>
<th>Book Name</th>
<th>No of Pages</th>
<th>% Error</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prateeksha</td>
<td>76</td>
<td>4.57</td>
<td>24.0</td>
<td>14.02</td>
<td></td>
</tr>
<tr>
<td>Birbal in Bahoshi</td>
<td>254</td>
<td>4.57</td>
<td>27.3</td>
<td>12.25</td>
<td></td>
</tr>
</tbody>
</table>

It can be seen that the recognition accuracy varies greatly. There are several reasons to which these errors can be attributed to like error in zone boundary identification, classifier error and error in text generation etc.. Fig. 8.1 and Fig. 8.2 give output of OCR software developed as a part of this research work.
8.3 Results and Discussion

Figure 8.1: Result of Gujarati OCR

(a) Page from Prateeksha
(b) Recognized Text
8.3 Results and Discussion

Figure 8.2: Result of Gujarati OCR
8.4 Conclusions and Future Work

Gujarati script is considered to be one of the most difficult scripts from OCR perspective. It poses greater challenges in terms of segmentation in absence of shirorekha and thereby increasing the number of classes.

In this research we have constructed the first end to end OCR system for printed documents of Gujarati script. We have presented various subtasks for this and also presented mathematical techniques that can be used to carry out each of these subtasks. The approach selected for this work is a zone separation based approach wherein we are segmenting connected components in each of the three logical zones and recognize them separately. The recognized glyphs are then used to generate text using finite state machine based model. This being a pioneering effort in building complete OCR system for Gujarati script, results of the work-flow suggested here are found to be encouraging for the future developments.

This work is being extended now as a part of the same project funded by the Ministry of Communications and Information Technology. One of the major experiments that can be done in future is to avoid zone separation and check if the classifier can be designed for such a large number of classes.