Towards e-Library of Telugu literature

Rajesh Babu Arja
(Roll no: 09305914)

Guide : Prof. J.Saketha Nath
Co-Guide : Prof. Parag Chaudhuri

Department of Computer Science and Engineering
Indian Institute of Technology Bombay

October 31, 2011
What we want to do?

- Digitalize Telugu literature

Why we want to do?

- For easy cataloging of documents
- To enable search on huge literature
- To preserve the literature of the Telugu
- To encourage transliteration of Telugu documents to other languages
How can we do?

- By Implementing OCR for Telugu
What is OCR?

Figure: Optical Character Recognition (OCR)
Applications

- Library cataloging
- Base for many NLP applications.
- Applications in banks and post offices.
Our Scope

Our scope

- Scanned Telugu text documents
- Documents with printed Text format
- Documents without any images

Challenges involved

- Resolution of document is not in our control
- Noise can be present in the scanned document
### Why Telugu OCR is difficult?

#### Telugu vs English

<table>
<thead>
<tr>
<th></th>
<th>English</th>
<th>Telugu</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of classes</td>
<td>52</td>
<td>more than 400</td>
</tr>
<tr>
<td>No. of connected</td>
<td>1</td>
<td>1-5</td>
</tr>
<tr>
<td>components per character</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dependency among</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>connected components</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Confusion Characters</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Table**: Comparison of Telugu and English languages
Confusion characters

Figure: Confusion Characters example
Why Telugu OCR is difficult?

Telugu vs Other Indian Languages

<table>
<thead>
<tr>
<th></th>
<th>Devanagari</th>
<th>Telugu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word Segmentation</td>
<td>easy</td>
<td>difficult</td>
</tr>
<tr>
<td>Confusion Characters</td>
<td>less</td>
<td>more</td>
</tr>
</tbody>
</table>

Table: Comparison of Telugu and Devanagari languages

<table>
<thead>
<tr>
<th></th>
<th>Tamil</th>
<th>Telugu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vowel modifiers</td>
<td>connected</td>
<td>disconnected</td>
</tr>
</tbody>
</table>

Table: Comparison of Telugu and Tamil languages
Comparison of Telugu OCR

<table>
<thead>
<tr>
<th>S.No</th>
<th>Features</th>
<th>Classification</th>
<th>Font Ind</th>
<th>Noise free</th>
<th>Open Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>Primitive</td>
<td>Decision trees</td>
<td>No</td>
<td>No</td>
<td>n/a</td>
</tr>
<tr>
<td>[2]</td>
<td>Circular</td>
<td>Template match</td>
<td>Yes</td>
<td>No</td>
<td>n/a</td>
</tr>
<tr>
<td>[3]</td>
<td>Wavelet</td>
<td>neural networks</td>
<td>No</td>
<td>No</td>
<td>n/a</td>
</tr>
<tr>
<td>[4]</td>
<td>Fringe map</td>
<td>Template matching</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>[5]</td>
<td>Gradient dir</td>
<td>Nearest neighbor</td>
<td>No</td>
<td>No</td>
<td>n/a</td>
</tr>
<tr>
<td>6</td>
<td>?</td>
<td>?</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table: Comparison of Telugu OCR literature
Comparison with state of art Telugu OCR

<table>
<thead>
<tr>
<th>S.No</th>
<th>Input</th>
<th>Output</th>
<th>Font Ind</th>
<th>Noise free</th>
<th>Open Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PGM</td>
<td>ACI</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>any image format</td>
<td>txt/pdf</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Our Goal

Our goal is to implement OCR with following features

- Works as font independent system
- Works on noisy data
- Capable of digitalizing documents in DLI.
- Open Source project
- Web interface for OCR
Work done in stage 1

- Implemented end to end system
- Explored various stages of OCR
- Implemented each stage with basic methodologies
- Observed the problems involved in each stage
- Identified potential improvements to do in each stage
Stages of OCR

- Binarization
- Skew detection and Correction
- Noise removal
- Segmentation of lines, words and characters
- Feature extraction and classification
- Rendering data
What is Binarization?
Binarization

Our Methodology:

- Using global thresholding
- Using Java Advanced Imaging API [1]

Results:
- Obtained more than 90% accuracy.
What is Skew Detection?

Figure: Skew Correction example
Skew Detection and Correction

Our Methodology:

- Using Hough’s transformation
- Using Java Deskew Implementation [2]

Results:

- Correcting skew angle of ±20.

Limitations

- Don’t work for multi-skewed document
What Segmentation?

Line segmentation example

Connected component segmentation
Why Segmentation?

**Advantages of Segmentation**
- To make processing easier in next stages
- To minimize number of classes

**Different types of segmentation**
- Line segmentation
- Word segmentation
- Connected component segmentation
Connected component segmentation

Our Methodology

- Found 8-way connected components
- Implemented similar algorithm as flood fill [3]

Figure: Connected components example
Why Line Segmentation?

- Characters are not labeled sequentially
- To bundle the consonant modifiers with corresponding characters
- To process document in order
Our Methodology

- We found each line consists of four states
- We found histogram of density of black pixels in each line
- Used Hidden Markov Model (HMM) for segmentation of lines
- Basing on state changes found out line boundaries
Example


Towards e-Library of Telugu literature
Observations and Future Work

Observations

- Works well with uniform spacing documents
- Non-Uniform spacing documents are not segmented properly

Planned Improvements:

- Planning to apply Conditional random fields (CRF) for line segmentation.
Feature Vector generation and classification

Our Methodology

- Using 0/1 feature vector
- Generated synthetic data for three Telugu fonts
- Using euclidean distance for classification
Feature Vector generation and classification

Observations
- Dividing the characters into groups will improve accuracy.

Experiments
- Found accuracy without classifying connected components into groups
- Found accuracy with classifying connected components into groups
Feature Vector generation and classification

Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Without grouping</th>
<th>With grouping</th>
</tr>
</thead>
<tbody>
<tr>
<td>0/1</td>
<td>33.75%</td>
<td>41.58%</td>
</tr>
<tr>
<td>SIFT</td>
<td>10.40%</td>
<td>15.97%</td>
</tr>
</tbody>
</table>

Table: Classification Accuracy statistics
Post Processing

Our Methodology

- Property file created which maps each character label to corresponding Unicode
- Unicode corresponding to classified character label is picked
- Identified the character position and additional care taken accordingly
Example

Figure: Example for rendered output text
Observations

- Word spacing is not completely perfect.
- Paragraph beginning with space are not identified.
- Headings with large font are not taken care
Noise

Sources of Noise

- Quality of printer
- Quality of scanner
- Age of the document

Advantages of Noise removal

- Improves recognition results.
- Facilitates better processing in next stages.
Noise Removal

Our Methodology

- Found connected components (CC)
- Identified three features for CC
- Extracted three features for all CC
- Applied Expectation Maximization (EM) algorithm for clustering
Experiments and Results

Experiments

- EM algorithm with two clusters
- EM algorithm with three clusters
- K-means clustering with two clusters

Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Noise Clusters removed</th>
<th>Text clusters removed</th>
</tr>
</thead>
<tbody>
<tr>
<td>EM 2-cluster</td>
<td>70%</td>
<td>0.01%</td>
</tr>
</tbody>
</table>

Table: Noise Removal statistics
Example

Before Noise removal

After Noise removal


Towards e-Library of Telugu literature
Example

Before Noise removal

After Noise removal


Towards e-Library of Telugu literature
Observations

- Underlines and hand written characters are not removed
- Non-Telugu characters are not completely removed
- In some documents border noise is not removed
- Joint Telugu characters are removed
Future Work

Planned Improvements:

- Including more features like thickness of character
- Including the structural features of character shapes
- Including position feature to remove border noise
- Trying to implement methods to find overlapping characters
Conclusion

- Created end to end functioning basic OCR.
- Binarization with global thresholding giving 90% above accuracy.
- Able to correct skew angle of ±20
- Able to remove above 70% noise from document
- Able to segment lines with more than 95% accuracy
- Able to classify characters with more than 30% accuracy
- Able to render the characters by finding matching Unicode
Future Work

- Consider more structural features for improving accuracy and better noise removal.
- Apply CRF for line segmentation
- Will use language models for correcting confusion characters and broken characters
- Improve classification models.
- Implement page layout analysis module
- Implement a web based Telugu OCR
References

- Java Advanced Imaging API.
  http://java.sun.com/javase/technologies/desktop/media/.
- Java Deskew by Hough Transformation.
- Flood fill Algorithm.
  http://en.wikipedia.org/wiki/Flood_fill
- WEKA Java API.
  http://www.cs.waikato.ac.nz/ml/weka/
- Digital Library of India.
  http://www.dli.gov.in/
- Drishti Telugu OCR.
  http://www.ildc.in/Telugu/htm/lin_ocr_spell.htm
References


Thank You