Automated Processing of Census Forms in Tamil

Shashi Kiran[1], Rituraj[1], Suresh Sundaram[1], Swapnil Belhe[2], A G Ramakrishnan[1]
[1]MILE Lab, Dept of Electrical Engineering
Indian Institute of Science, Bangalore 560 012.
[2]Center for Development of Advanced Computing, GIST Group, Pune
shashi.reach@gmail.com, sureshsundaram1980@gmail.com
[2]swapnilb@cdac.in,ramkiag@ee.iisc.ernet.in,

1. Abstract

This paper describes automatic form filling system for collecting census data based on online handwritten character recognition for Tamil. The aim is to facilitate easy digitization of Indian language ink-data gathered from field. The application interface is designed in such a way that the same application can be adopted for other Indian languages. It also describes the common interface framework required to facilitate this multiple language recognition engines. For Tamil or any other Indian scripts; inputting isolated symbols is not practical, hence the application uses non-isolated character recognition. The application incorporates simplified method of form design, layout analysis, engine error correction; engine level limited vocabulary based post-processor, validations etc. The performance evaluation of this application is also carried out against the traditional methods and promising results are obtained. The same system can be easily adopted for other types of forms required to be filled in Tamil e.g forms used by Government institutions, Banks etc.

2. Introduction

India is one of the very few countries in the world, which has the proud history of holding census every ten years uninterruptedly since 1872. The census provides information on size, distribution and socio-economic, demographic and other characteristics of the country's population. The data collected through the census are used for administration, planning and policy making as well as management and evaluation of various programs by the government, NGOs, researchers, commercial and private enterprises, etc. [1]

Government agents visit each and every household in the country and collect complete data about the people and the condition of their houses. These agents gather this data by filling printed forms. The filled forms are then used to manually enter data in computers. This procedure takes long time and is prone to human errors while re-entering data manually. The data collection and the data storage are the two basic stages of this activity.

This census form processing system requires the members to carry the digital pads to the site, collect the information on this device, once the data collection is done plug in this digital pad to the computer and this application will generate the database.

Gathering information by using printed forms is predominantly used in governmental, educational, banking domain. Even on-field surveys are carried out using local languages. Traditionally, the major

surveys like census are conducted on field by pen & paper wherein a printed form is filled by a surveyor. The surveyor collects information on the printed forms; and when the data collection is completed, forms are first scanned and then send for the verification and data entry. This method works well for small surveys requiring limited information. But when the survey requiring detailed information is to be carried out across cities, states and country; the traditional method becomes time consuming. The large amount of time is spent on scanning, verification and data entry.

The goal of this automated census form filling system is to provide simplified form filling process which works with very low cost digitizers and considers major Indian languages. Reduction in cost of digitization is also considered during design of this system.

In this system, the data collection process remains the same except an offline digitizing tablet is used for writing forms. The paper is kept on the tablet and the user writes on the paper by using special pen. The currently available offline digitizing tablets provide the advantage of retaining hard-copy of the filled forms for future use. These tablets with no display capture the ink data as X-Y co-ordinates and pen pressure. This collected ink data in X-Y co-ordinate format is then submitted to the form processing application which converts it to computer editable text.

3. Data Collection

The data collection phase is most crucial phase of the form processing application. The procedure for collecting data using digital tablet is similar to what currently being followed by data collectors. Instead of only-paper based forms the battery backed digital tablet is used along with digital ink based pen as shown below.

The data collected on such digital device can then be downloaded on the Computer for further processing. This is where Tamil online handwritten engine come into effect.

The Surveyor who is collecting the data is required to follow the manifest for collecting data which is similar to traditional data collection. We have collected data from ten writers of Tamil for testing by using G-Note 7000 digitizing tablet working at 160 points/sec.

4. Architecture of System

This system is primarily divided into four modules for simplicity during development, the modules are,

- User Interface
- Input Data Extraction module
- Recognition Engine Interface module
- Database Generation module

The application architecture is kept broad to easily plug-in different components. The application is currently designed to support two different types of digitizers namely Genius G-Note and iBall TakeNote but can easily be adopted to other devices. The Genius device produces the files with TOP extension while iBall produces files with DHW extension. As explained in the previous section the raw ink data (x, y co-ordinates) coming from the device is given to the application as input. Figure 1 explains the broad architecture of the application.

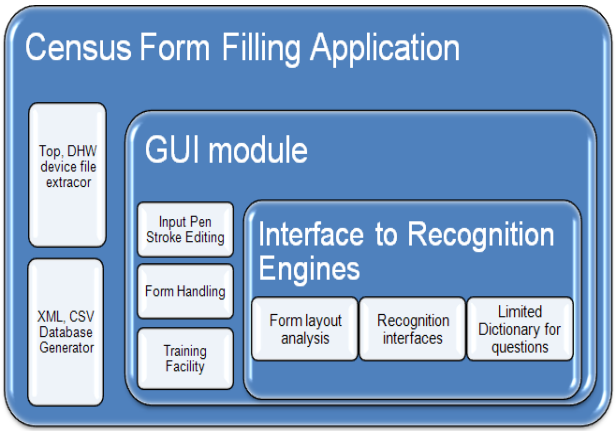

Figure 1: Architecture of Tamil Form Processing Application

The input data is displayed in full page format for verification. If the data contains any abnormal and misaligned ink strokes, they are removed by the operator. The verified data is then fed to the layout analysis module. This module aligns the input data to the pre-defined template using positional analysis. This way, the answers by the client are mapped to corresponding questions. Once the correct alignment is achieved the text/numeral fields are separated from ticks. The text and numerals are given to Indic engines for recognition. This is achieved by using common interface framework used to link the Tamil handwritten recognition engine to the application. The output of the engine contains the recognized text with confidence scores. The confidence score is used to provide the operator with choice to select the correct outcome. If the score is high enough then no selection choice is provided.
The operator is also provided with easy Tamil text editing functionality like virtual keyboard.

5. Common Interface Framework

Since this form processing system is designed to accommodate all Indian languages along with Tamil, there are many recognition engines to be interfaced to the system; hence there is a great need to have a standard communication protocol between different handwritten recognition engines and census application. A common Interface framework provides this vital link between the recognition engines and the application. This framework provides the flexibility to add/remove recognition engines. Also an entire new application can be built around this framework without the need of knowing the intricacies of online character recognition technologies. So this framework provides the scalability of adding new engines to the existing application. Simply put, it act as messenger between the application and the engine and thus helps to change any layer on the fly (i.e. we can change the application without disturbing the rest of setup or vice versa).

6. Pre-processing

6.1 Automated Segmentation

The segmentation of the validated data reduces the overhead of the Tamil recognition engine and helps in fast and accurate recognition of the handwritten text. All strokes in a page are given to segmentation module. In this module, all the strokes are arranges as per their position in a page irrespective of their order of occurrence. This module segments the lines based on horizontal projection of the strokes. Once the lines are segmented, the gaps between histogram (vertical projection) of each stroke on the line are clustered into within-word-gaps (WWG) and between-word-gaps (BWG) [2]. The words are separated by WWG.

6.2 Form Cleaning

The digital ink data collected by the surveyor often has noisy, unclean ink. Sometimes, the surveyor himself writes some notes, annotations, comments on the pages inside or outside the page boundaries. There also could be scrubbing which is large enough, crossing the full page or questions by digital ink. There could be overwriting, page misalignment or completely missing the pages of the form. This kind of data is required to be cleaned before proceeding for recognition. Otherwise the recognition accuracies could be very poor. Since it is difficult to clean all such noise automatically, the form cleaning is done semi-automatically. The cleaned, verified and segmented data is passed for recognition by the language specific engines.

7 Additional Features

7.1 Tick detection

The forms used for census survey contains various data fields like text, numerals, check boxes etc. The surveyor is supposed to tick into the boxes wherever necessary. First, the check boxes needs to be separated from other fields. This is achieved by positional analysis of the form. Once the relative position of the ticks is identified, it is required to separate actual tick marks from the scrubbing and other unwanted strokes. The tick detection is used for detecting the check mark on the form; this includes fields like radio buttons or check boxes. There are various ways of recognizing ticks in this implementation we have used Dynamic time warping (DTW). It is an elastic matching algorithm for matching the similarity between two given sequences. The similarity matching is based on the distance measure. The sequence with minimal distance is considered for optimal match. Since the distance based sequence matching is not feasible for every point in sequences we need to put two conditions i.e. boundary and continuity condition. The continuity condition decides how much the matching is allowed to differ from linear matching [3, 4]. The Boundary condition states that first and last points of sequences will be matched with each other. The continuity condition decides the measure of elasticity given by the formula:

$$\frac{N_2}{N_1}i - cN_2 \leq j \leq \frac{N_2}{N_1}i + cN_2$$

where *c* is the continuity constant, N_1 and N_2 are the number of points in first and second curve respectively. The points *i* and *j* of the first and second curve respectively can be matched only if above condition is satisfied. For *c=0* the resulting match is same as linear matching.

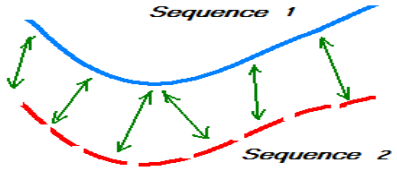

Figure 2 DTW matching of two ticks

DTW is trained on data collected from 20 clients, for ten different styles of writing ticks. Certain conditions are applied to the data before passing the data to DTW such as checking for number of strokes.

7.2 Error Correction Measures:

The recognition accuracy of the underlying Tamil handwritten recognizer is not 100% hence it becomes responsibility of the application developer to highlight those inaccuracies to the *operator* in a mild way. The ideal application for such kind of systems would be one which not only gives the operator the helping hand in case of a recognition error but also trains the engine over time by understanding corrections made by operator. It also allows editing the input file with stroke addition/deletion functionality and saving the same.

Significant percentages of operators are bound to get irritated in case of repeated errors on part of recognition (governed by recognition engines). An effective user interface in this case would be able to minimize the edits required by use of automatic focusing, multimodal inputting, easy suggestion list etc. The application is built with stroke correction facility. If the ink data contains spelling mistakes or ambiguities then the operator can re-write the strokes and save it for future use. The operator is also given the facility to type using virtual keyboards. The complete application is localized for Tamil.

Since each field is separated by the values it may have, like text, numerals, numerals with symbols etc. The application performs the broad validity check on each output returned by the engine. This helps in narrowing down the errors for each field.

Based on the probability measure returned by the Tamil recognizer engine, the application takes the decision on whether the recognized text is suitable for displaying. If not, the field is marked with red color pointing to the operator about possible error that may require relook.

7.3 Standardized output format:

The operator is given the flexibility to save the recognized data in Unicode based Comma Separated Variable (CSV) files or Extended Mark-up (XML) files. These files can be easily imported by any of the database engines like Oracle, SQL etc.

7.4 Wordlist for Post-Processing

The common interface framework includes the interface to send the domain specific Tamil wordlists to the handwritten recognizer. The application developer changes the wordlists as per different requirements of the applications and sends it to engines so that language models can be applied in order to improve the performance. The structure of the wordlist is defined by the framework.

8. Results and Discussions

We compared the performance of this automated census form filling application with the traditional ways of conducting census surveys. The performance of the application is highly dependent on the recognition accuracy of the Tamil Online handwritten recognizer. Application uses some domain specific knowhow of census form data to improve on the base recognizer performance. In the traditional surveys, the filled forms are scanned and then manual data entry is done. In this experiment the census form used for evaluation was very comprehensive spanning over 10 pages. Following table shows the composition of the form used for the survey.

Field Type	Number of fields per form
Text (with wordlist)	8
Text (without wordlist)	17
Numerals	22
Check boxes (multiple ticks)	10
Radio boxes (single ticks)	27

The text fields like clients education background, spoken language, city, district, state etc. were backed with limited dictionary while fields like name, last name, address etc. were without dictionary.

For evaluating performance of traditional census data entry, the forms were given to 3 data entry operators who were regulars in Tamil typing. Total data was collected from 10 writers; each form contained 10 pages. The same forms are processed through this application. Following table shows the comparative results of traditional manual data entry and online character recognition based application.

	Average Time (Time/Form) (10 pages per form)	Average Error in output
Manual Data Entry	89.09 *sec.*	2.68 %
Automated Form processing + Verification	30.37 *sec.*	13.88 %

As seen from the above table, the manual data entry took more time but numbers of errors were less. Note that the page scanning time required for manual data entry is not considered in the above table.

9. Conclusion & Future Work

In our experiment, the online handwritten character recognition based form processing for Tamil clearly showed a promising area for further research especially where collecting huge quantities of data from field and converting it into editable text is concerned. In this application we retain the paper based form filling & editing thus allowing more natural text inputting and reduce cost incurred on display based inputting devices. In future, we would like to study the impact of the Indian language word models on user acceptance of online handwriting recognition.

10. Acknowledgment

The authors would like to thank Consortium for "Online Handwritten Character Recognition" and Technology Development for Indian Languages (TDIL), Department of Information Technology (DIT), Government of India for funding this consortium project. The authors would also like to thank all the consortium chief investigators and members consisting of IISc-Bangalore, ISI-Kolkata, IIT-Madras, IIIT-Hyderabad, CDAC-Pune for their valuable inputs.

References

1. Ashish Krishna, Girish Prabhu, Kalika Bali, Sriganesh Madhvanath, "Indic scripts based online form filling - A usability exploration", 11th International Conference on Human-Computer Interaction (HCI), Las Vegas, 2005.
2. Soo H. Kim, S. Jeong, Guee-Sang Lee, Ching Y. Suen, "Word Segmentation in Handwritten Korean Text Lines Based on Gap Clustering Techniques", Proc. Sixth International Conference on Document Analysis and Recognition (ICDAR-'01), Seattle, WA, pp. 189-193.
3. Ralph, Niels and Louis, Vuurpijl, "Dynamic Time Warping Applied to Tamil Character Recognition," Proc. Eight International Conference on Document Analysis and Recognition (ICDAR'05), pp. 730-734.
4. N. Joshi, G. Sita, A. G. Ramakrishnan, and S. Madhvanath, "Comparison of elastic matching algorithms for online Tamil handwritten character recognition," Proc. Ninth International Workshop on Frontiers of Handwritten Recognition (IWFHR'04), pp. 444–449.
5. XStroke: Full-screen Gesture Recognition for X, Carl D. Worth Information Sciences Institute University of Southern California Arlington.
6. UPX- The best from UNIPEN and ink ML, *http://unipen.nici.kun.nl/upx/, 2002.*
7. Swapnil Belhe, Srinivasa Chakravarthy, A.G. Ramakrishnan, "XML Standard for Indic Handwritten Indic Database" Proc. International Workshop on Multilingual OCR (MOCR-09), Barcelona, Spain, July 2009.