

1. Development of OHWR System for Hindi

Swapnil Behle & Team
C-DAC, Pune

Introduction:

Handwritten Recognition poses a special problem in relation to Indian scripts. Most Indic scripts use a large number of characters, as opposed to English. Each handwritten script admits various handwriting styles that vary according to region, making recognition a challenging task. Thus for all scripts; the writing style varies from North to South India also from person to person and age group to age group. Hindi, which is one of the most widely used language in India is written in many states like Uttar Pradesh, Delhi, Madhya Pradesh, Rajasthan and many others. The Hindi language and the Devanagari script is widely used in India in Schools, Colleges and Government offices.

In vast country like India, the Government-to-people interaction happens in writing. Most of the time, people are required to fill out some kind of form. The data collected through such forms 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. The forms are filled either in English or local languages and majority of them are handwritten. 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.

Under this project the application is getting developed which will require the members to carry the digital tablets to the site, collect the information on this device, once the data collection is done plug in this digital pad to the computer or wirelessly send the data to centralize server. The handwritten data will semi-automatically get converted to editable text. This will reduce the manual data entry operation, and make this activity much faster.

1.0 Important Milestones Achieved

- Implemented new approach of classification using Convolutional Neural Network for 42 classes. Total of 33197 and 8321 train and test examples are used for experimentation. Results show 99.6% and 96 % on train and test data respectively.
- Developed and published a new Handwritten Word Recognition scheme using HMM and Symbol tree in DAR 2012. The proposed approach performs recognition of online handwritten isolated Hindi words using a combination of HMMs trained on Devanagari symbols and a tree formed by the multiple, possible sequences

of recognized symbols. Tests performed on 10,000 words yield an accuracy of 89%.

- A new Lite version of Devanagari recognizer for limited classes is also developed especially targeted for mobile devices. This recognizer is suitable for limited inputting over small screen sizes.
- Various devices with stylus based inputting support were extensively tried. This activity is pre-cursor to actual data collection and training efforts.
- Developed Census Form Processing application for Tablet PC.
- A software design document is prepared for generic form filling application. This application allows the use of underlying OHWR engine for conversion of online data into editable text. Apart from this, this application also performs comprehensive repository management, searching within data, statistics generation etc.
- Development of new XML schema supporting sub-strokes and sentence level data annotation has been designed and circulated to all members.
- Modification of Application Interface Framework to caters to Text Block recognition as well while maintaining the backward compatibility. The document of this new Interface Framework is circulated to all members.

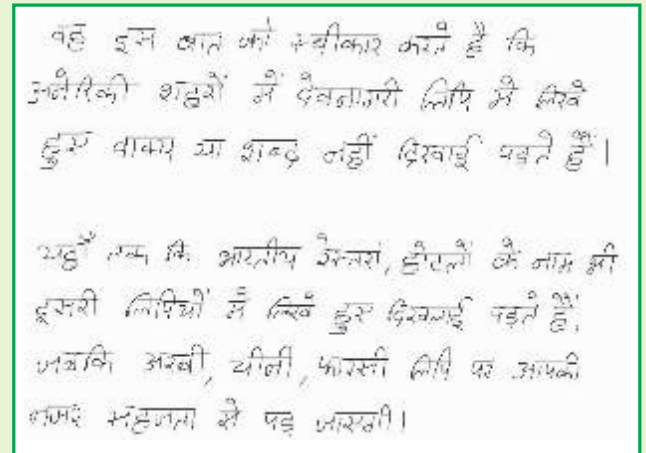
2.0 Design and Development of Newer Approaches

Improvements in existing engine:

The Hindi recognition engine developed during initial work on stroke level with no facility to dynamically train the

engine. Later on this training facility to the user whereby the engine will get trained as per user inputting is envisaged. The sentence level recognition is also being attempted.

A typical sentence level text in Hindi is shown below :



2.1 Convolutional Neural Network (CNN)

Convolutional Neural Networks are a special kind of multi-layer neural networks. Like almost every other neural networks they are trained with a version of the back-propagation algorithm, where they differ is in the architecture. Convolutional Neural Networks are designed to recognize visual patterns directly from pixel images with minimal pre-processing. They can recognize patterns with extreme variability (such as handwritten characters), and with robustness to distortions and simple geometric transformations [1]. In CNN, shift and distortion invariance is automatically obtained by forcing replication of weight configuration across space and by using subsampling layer. Convolutional network force the extraction of local features by restricting the receptive field to be local.

Limitation of Neural Network

- There is no inbuilt invariance with respect to translational or local distortion of the input.
- Topology of the input is entirely ignored.

Advantages of CNN

Local receptive fields

To extract elementary visual features such as endpoints, edges, corners etc.

Shared weights

Units in a layer are organized in planes within which all the units share the same set of weights. These planes, also called feature maps, of simple units called neurons. This characteristic is justified by the fact that an elementary feature detector useful on one part of the

image is likely to be useful on the entire image.

Spatial or temporal subsampling

Reduces resolution of feature map and reduces the sensitivity of the output of shifts and distortions

2.1.1 Architecture of CNN

The first layer has only one feature map which is the input image itself. In the following layers, each feature map keeps a certain number of unique kernels (2D arrays of weights), equal to the number of the feature maps in the previous layer. The size of each kernel in a feature map is the same and is a design parameter. The pixel values in a feature map are derived by convoluting its kernels with the corresponding feature maps in the previous

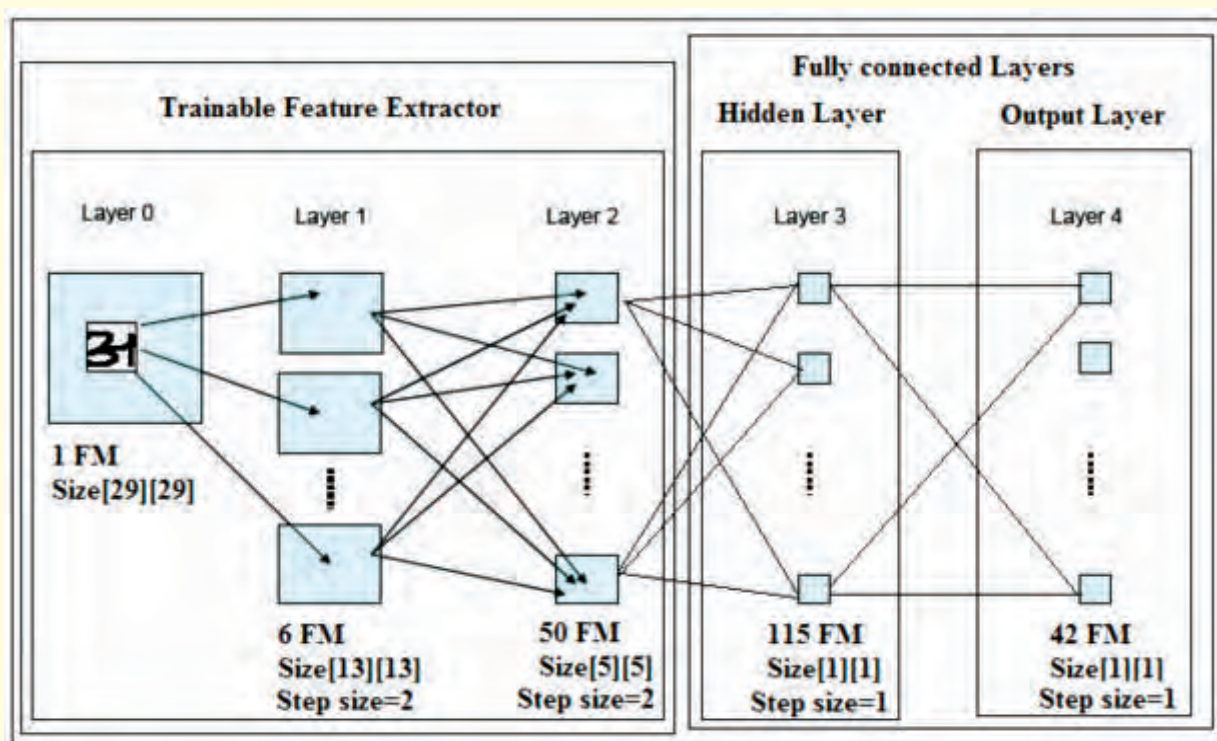


Figure 2.1 Architecture of CNN FM = feature map

layer. The number of feature maps in the last layer is equal to the number of output classes. For example in 0-9 digit recognition, there will be 10 feature maps in the output layer, and the feature map with highest pixel value will be the result.

2.1.2 Implementation details

Convolution: Convolution of a (N×N) image with a (K×K) kernel can be understood by sliding a (K×K) window over the input image iteratively. For each position of the window, an output pixel is generated by taking the dot product (sum of the multiplication of the corresponding pixels) of the kernel with the input pixels lying under the window. In the following figures we have shown the calculation of first two pixels of the output image Y, generated by convolution of a (6×6) input image X with a (2×2) kernel W.

It can be noted that the size of the output image obtained by convolution of an N square input image to a K square kernel is (N-K+1) square.

We calculate the pixels which will be retained after applying the sub-sampling. This is equivalent to setting a step-size, which the convolution window in Fig.2 will jump both in horizontal and vertical directions instead of jumping by one pixel, as is the case with general convolution shown in Fig.2.2 this step size is indicated under each layer in the fig.2.1.

Forward Propagation: At a convolution layer, the previous layer's feature maps are convolved with learnable kernels and put through the activation function to form the output feature map. Each output map may combine convolutions with multiple input maps [2]. In general, we have that

$$x_j^l = f \left(\sum_{i \in M_j} x_i^{l-1} * k_{ij}^l + b_j^l \right)$$

where M_j represents a selection of input maps. Each output map is given an additive bias b , however for a particular output map, the input maps will be convolved with distinct kernels. That is to say, if output map j and map k both sum over input map i , then the kernels applied to map i are different for output maps j and k .

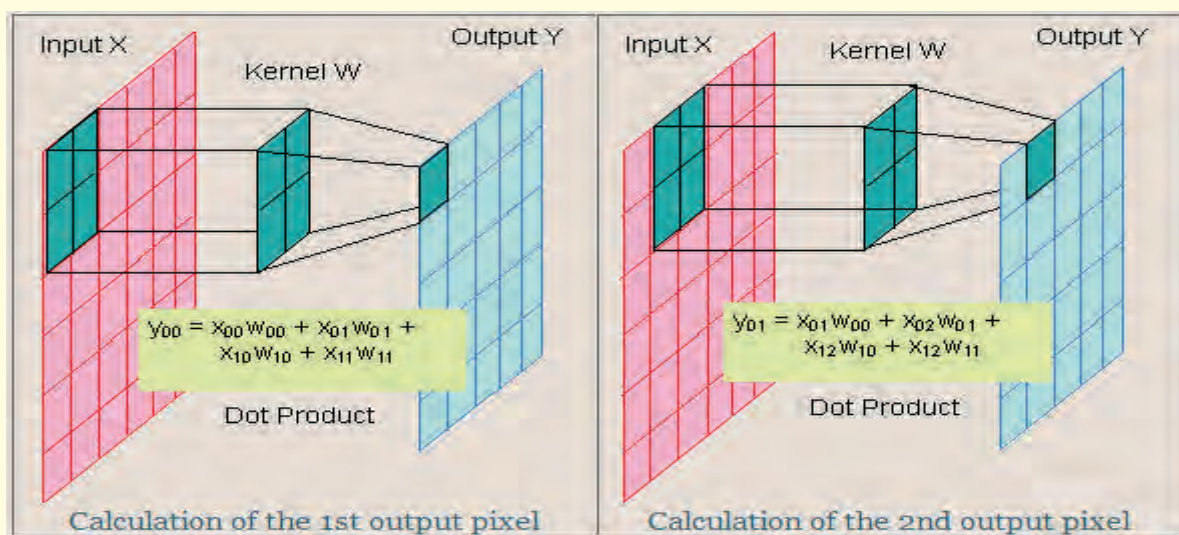


Figure 2.2 Convolution

Back propagation:

Back propagation is needed for training the CNN. The error in the output, i.e., the pixel values of the last layer, is propagated back through the layers and their weights are adjusted [3].

1. First we find the derivative of the error w.r.t. neuron values in the last layer

$$\partial X = X - X_{ideal}$$

2. The derivative calculated this way is w.r.t. the output that was calculated after applying the activation function. We get the error w.r.t. the “output before applying the activation function” by applying an inverse activation function “logit” to the output and multiplying it to the derivative calculated above.

$$\partial Y = \log it (X) * \partial X$$

3. Next we calculate the derivative w.r.t. weights. All the neurons in the previous layer that shared a particular weight during forward propagation would be used for

calculating derivative of the error w.r.t. that weight. The same kernel also connects all other pixels in layer J+1 to the other pixels in the same FM in layer J, and all those connections will contribute their share in the same way in calculating total derivative w.r.t. weights for this kernel. The same procedure is repeated for all other kernels belonging to each FM in layer J+1. Bias is added to each pixel of a FM during forward propagation, without having a connection to the previous layer. Therefore, error w.r.t. bias is calculated simply by summing up derivative of the error w.r.t. neuron values of all pixels in the FM.

4. After calculating derivative w.r.t. weights, we can update the weights using the following formula:

$$w_{new} = w_{old} - \alpha * \partial w$$

where α is the learning rate, and a very important design parameter.

Back Propagation is a robust algorithm for difficult connectionist learning problems. But

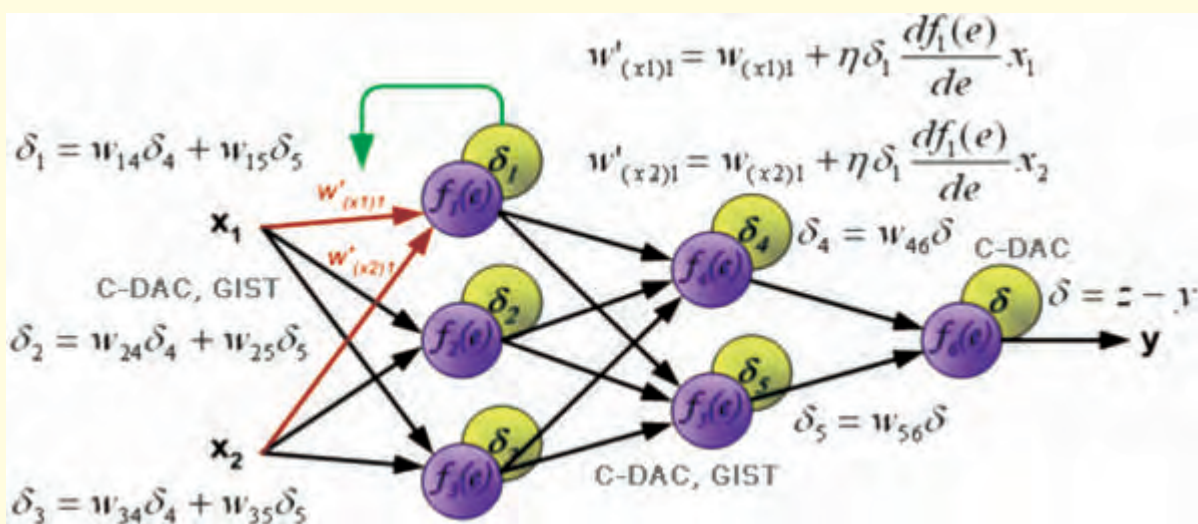


Figure 2.3 Back propagation

$$C_c(w_c + S) = C(w_c) + \nabla C(w_c)^T S + \frac{1}{2} S^T \nabla^2 C(w_c) S$$

the above explained Gradient Based Back Propagation Algorithm converges slowly. To counter this issue, a new approach makes use of the 2nd order derivative terms for reaching optimized weight values [4].

As per Taylor Series Expansion: where 'C' denotes the Cost function whose 'minimum value state' corresponds to the optimized weight Values.

At the Optimum weight Value state, the Derivative of the Cost Function is '0'. There at, value of the 'step' is as below:

$$S = -\nabla^2 C(w_c)^{-1} \nabla C(w_c)$$

So, the previous 'step size' or weight Update' of:

$$\nabla w_{ij} = -\epsilon \frac{\partial C}{\partial w_{ij}}$$

Now becomes:

The Second Order methods, with the above given Pseudo-Newton Step, converge

$$\nabla w_{ij} = -\epsilon \frac{\frac{\partial C}{\partial w_{ij}}}{\frac{\partial^2 C}{\partial^2 w_{ij}}}$$

faster than simple Gradient Based methods by talking into account the additional information about the objective function.

2.1.3 Dataset used for training

A database of handwritten patterns was prepared at CDAC Pune. It has a training set of 33197 and test set of 8321 examples consists of 42 root character classes for Devanagari.

Training Observation:

We plotted back propagation progress as a function of epoch. At each epoch, I plotted the learning rate eta, the MSE (of the training set), here are the results shown in figure 2.4.

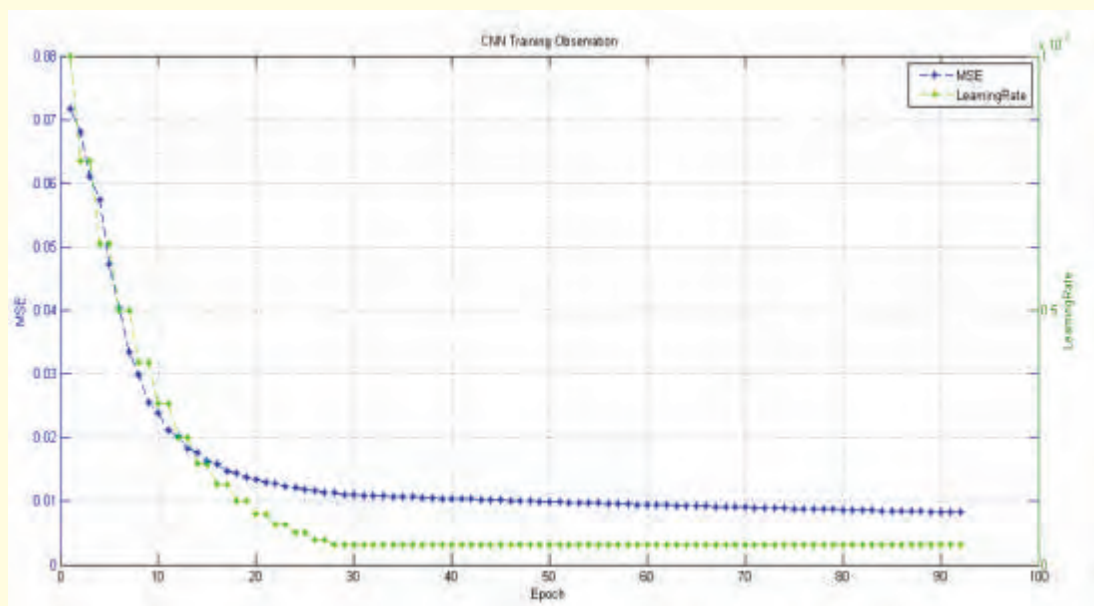


Figure 2.4 MSE and Learning rate vs. Epoch plot

Similarly, we plotted forward propagation progress as a function of epoch. At each epoch, we plotted the error rate (of the training set), here are the results shown in figure 2.5.

We have trained the classifier on 22758 examples. The error is close to about 0.392%, accuracy on above data is 99.6%. The second order back prop is done on 500 random

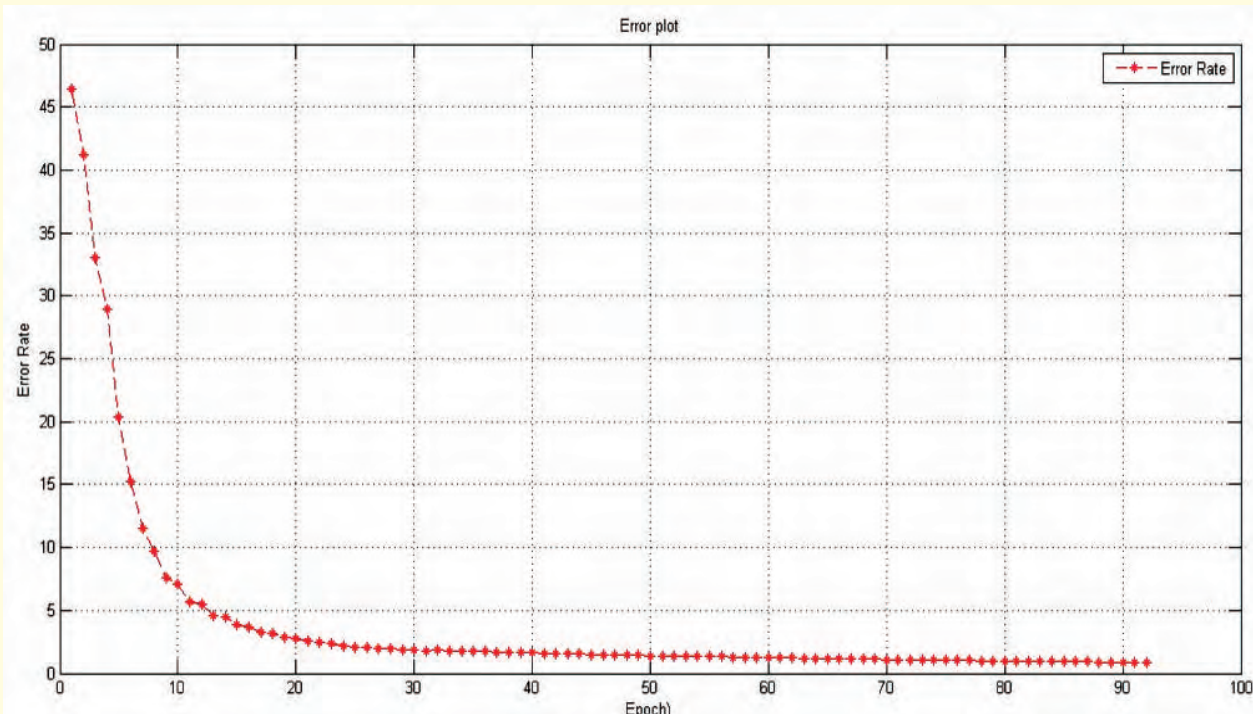


Figure 2.5 Error rate vs. Epoch plot

Based on above observation, training and test accuracy are as follows:

CNN	Accuracy
Trained Data (22758 examples)	99.6%
Test Data (12134 examples)	96%

Dataset

As shown in fig 2.6 dataset consists of various types of unusual patterns. CNN is able to identify such data which makes it invariant to translation, rotation and scale. Also it is invariant to any number of strokes such that u can write a root character in any number of strokes as long as it is visually identifiable.

samples so as to speed up the training.

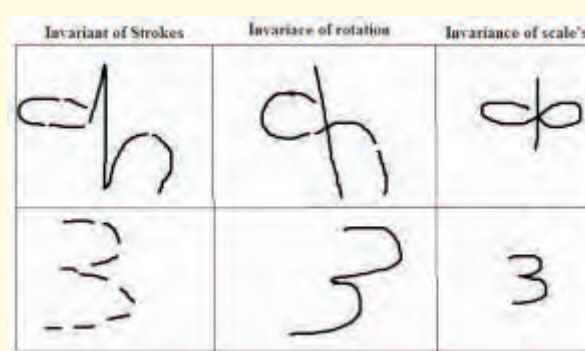


Figure 2.6 Weird styles recognized through CNN

2.2 Hybrid Character Segmentation

Character Segmentation is necessary module before character classification. We

have trained CNN for 42 classes. CNN recognizes segmented symbols from word. This symbols are basic unit for CNN. Symbol (Root Akshara) does not contain matras and shirokekha. Character segmentation is hybrid (online + offline) approach for segmenting symbols from handwritten word.

Shirokekha and Matras Separation are first step. Shirokekha is identified as upper horizontal line. Sometimes it cannot be top most because of upper matras Fig 2.7. To overcome this problem we identify matra strokes and non matra strokes using SVM classifier. Using geometrical rules we identify Shirokekha and matras. Then Root Aksharas are separated using geometrical information and rule based on spatial information.

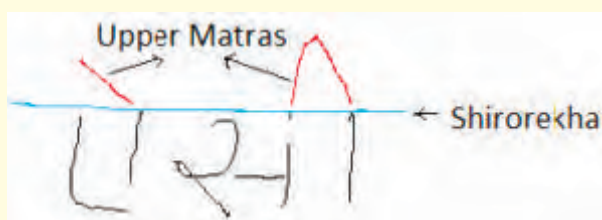


Figure 2.7 Word with Shirokekha and upper Matras are highlighted

Input	Output

Figure 2.8 Character segmentation

2.3 Handwritten Word Recognition using HMM and Symbol Tree

Introduction:

Major problem in recognizing of Handwritten Devanagari word is the segmentation of Akshara. It is the most

difficult part as Devanagari is a very complex script. Words are written in three Layers - Upper, Middle and lower as can be seen in Fig.2.9. Word can be having matras, conjuncts, Shirokekha(baseline). Besides, matras can appear in any region.



Figure 2.9 Devanagari Word

For the handwritten word, segmentation is more difficult than printed word (OCR). Handwritten data is more complex (as shown in Fig.2.10) because Aksharas sometimes overlap with each other. Hence they are not vertically separable like in OCR. Characters and matras may overlap between layers. Layer boundaries are not uniform through a word.

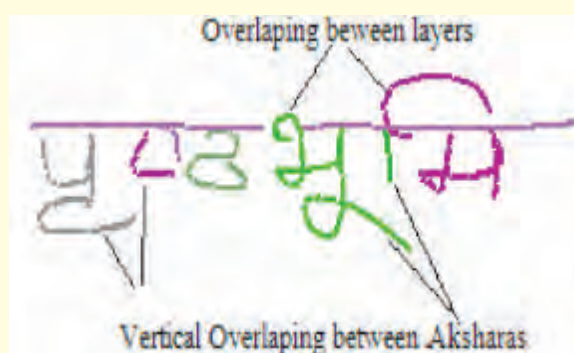


Figure 2.10 Overlapping in Handwritten Text

Here we proposed one method to overcome such problem in online HWR.

Training:

Online handwritten data contains series of

31 31 31

We observed one symbol is formed by maximum sequence of three strokes. We found there are 176 types are strokes (S) frequently used to Devanagari script. $S \in \{a, 2...176\}$ and there are 150 Symbols (Smb).

$$Smb \in \{ क, ख, ग, ... ञ, टि, ... ड, च, ... न \}$$

$$P(S_t), P(S_t|S_{t-1}), P(S_t|S_{t-1}, S_{t-2})$$

symbol. We compute probabilities using SRILM for each symbol. We called this model as **Stroke Model**. Finally we are having Stroke Models for all 150 symbols.

Word contains sequence of strokes. The problem is to segment this word into meaningful symbol Unicode. Segmentation and Recognition process are running simultaneously. We form a tree having three branches as shown in Fig.2.12. Each node is having maximum likelihood Symbol with probability (P) where $i=1, 2$ or 3 .

$$Smb_1 = \underset{Smb}{\operatorname{argmax}} P(S_t)$$

$$Smb_2 = \underset{Smb}{\operatorname{argmax}} \text{ P}(S_t | S_{t-1})$$

$$Smb_3 = \underset{Smb}{\operatorname{argmax}} \text{ P } (S_t \mid S_{t-1}, S_{t-2})$$

Suppose stroke sequence with length 3 having one stroke of next Symbol then maximum likelihood symbol will have low probability i.e. Current symbol is made by 1st two strokes only. If maximum probability (P) of stroke sequence in likelihood symbol (S_{mb_i}) is less than Threshold (T) then we are not adding further node to the tree. After forming tree, travel from root node to leaf node having probability greater than threshold (T). Finally sequence of symbols travelled is nothing but recognized Word.

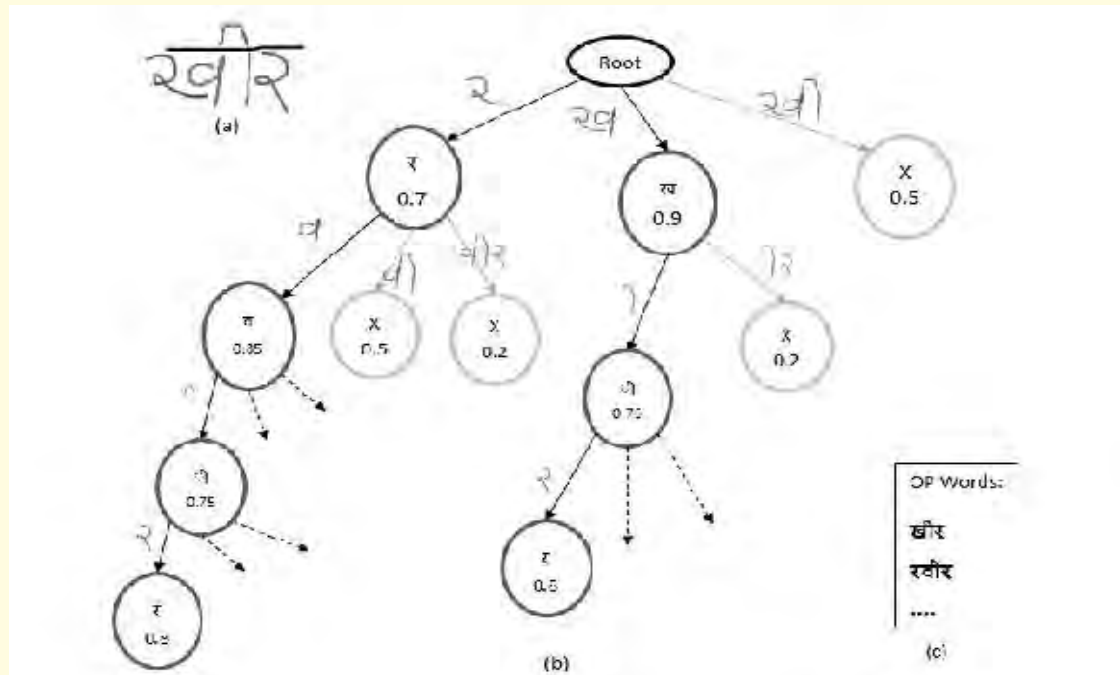


Figure 2.12 Building the Symbol Tree

Output

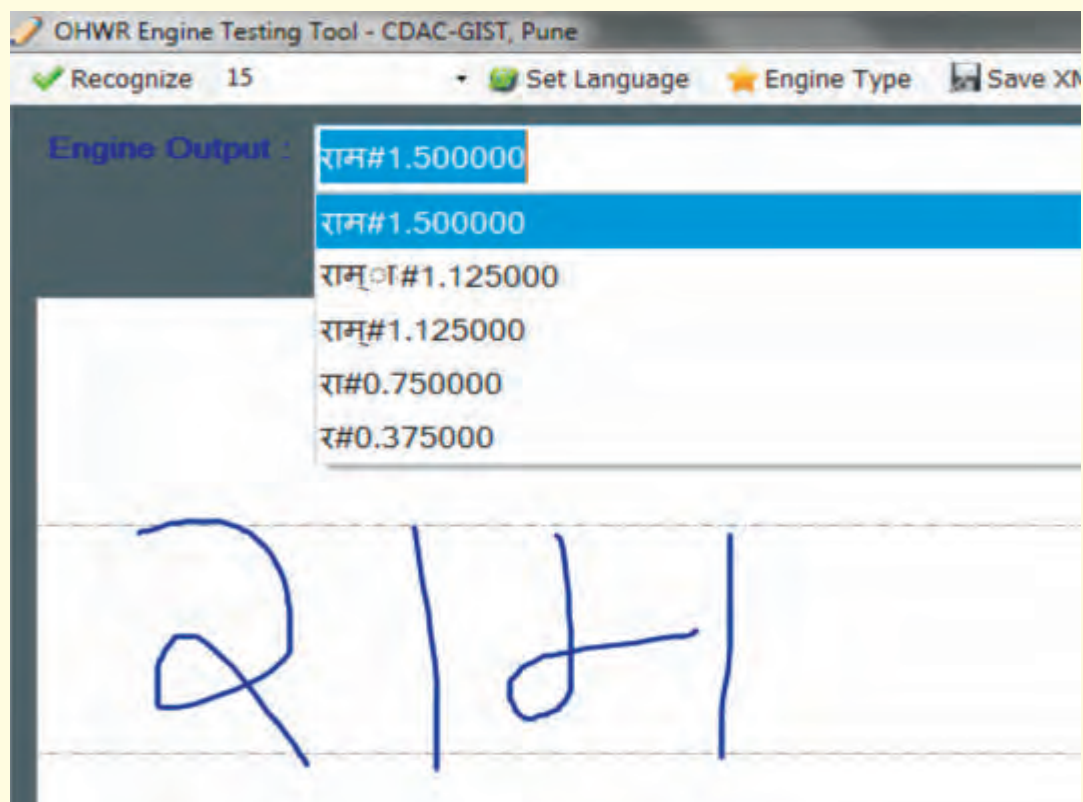


Figure 2.13 Application Output

Performance and Evaluation:

Tests performed on 10,000 words yield an accuracy of 89%. The word level accuracy is found to saturate from a maximum stroke group size of 3 to a maximum stroke group size of 5 (at 89%), while the time required to build and traverse the tree increases steadily. Thus, the configuration with a maximum stroke group size of 3 has been chosen for the actual recognition system. This has an average execution time of 330 ms/word, assuming an average of 10 strokes per word.

Our approach fails if the temporal sequence of strokes is not maintained and an out of order stroke appears in the sequence. There are very few such cases in our data and they were not considered for evaluation.

Future Work:

Our future endeavour will be to improve the performance and speed of the recognition system, keeping in view the mobile devices, by making use of Convolutional Neural Networks (CNN). Also, we intend to include a language model (LM) in order to filter the erroneous outputs and narrow down to one.

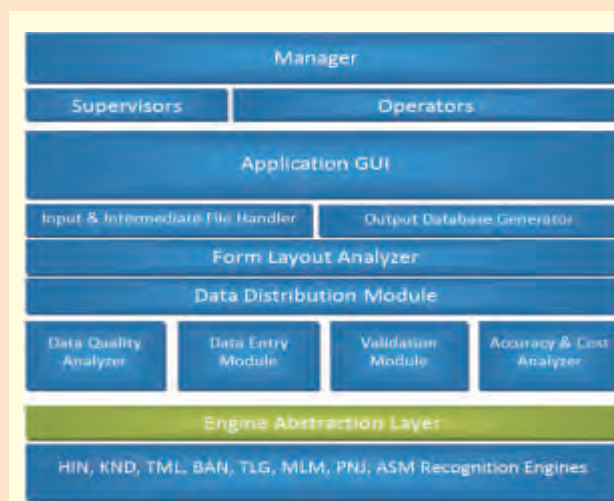
3.0 Form Processing Application and Basic SDK

Basic Architecture of Form Processing System:

This application contains following main components,

- User Interface
 - Input file handler module
 - Form Layout Analyzer

- Data distribution module
- Database Generation module
- Engine Abstraction layer



The form layout analyzer module is developed for basic forms. The form designer application for designing various forms is also developed.

The recognizer engine is packaged in the form of an SDK and basic pilot deployment is to be started for different on-field scenarios like mobile based application for farmers in UP, Form filling application for submitting new Ration card applications, form filling for bank account opening etc. The feedback from these applications will be collected in order to improve the recognition accuracy.

4.0 Census Data Collection Application (Tablet-PC Edition)

The census data collection application is now designed for tablet-PC.



Following are the screens of the Census application for Tablet-PC

Census Data Processing Application (CDAC - GST, Pune)

File Edit Tools Options Help

CDAC हिंदी

Page 1 Page 2 Page 3 Page 4 Page 5 Page 6 Page 7 Page 8 Page 9 Page 10

Q 1.1 पहला नाम राम

Q 1.2 मध्य नाम/ पिता का नाम/ पति का नाम अनिल

Q 1.3 उपनाम/ परिवार का नाम देशमुख

Q 2 मुखिया से संबंध बेटा

Q 3 लिंग ☐ स्त्रीलिंग ☒ पुलिंग

Q 4 पिछले जन्मदिन की उम्र (पूर्ण वाक्य में) २९

Language: HIN Form Language: HIN NUM

Census Data Processing Application (CDAC - GST, Pune)

File Edit Tools Options Help

CDAC தமிழ்

Page 1 Page 2 Page 3 Page 4 Page 5 Page 6 Page 7 Page 8 Page 9 Page 10

Q 1.1 முதல் பெயர் சண்முகம்

Q 1.2 நடுப்பெயர்த்ததை பெயர்க்கணவர் பெயர் ரெகாபாத்ரி

Q 1.3 பட்டப்பெயர்/குடும்பப்பெயர் பள்ளம்

Q 2 தலைவருடன் உள்ள உறவு மகன்

Q 3 பாலினம் ☐ பெண் ☒ ஆண்

Q 4 கடைசி பிறந்தநாளன்று வயது (முழுமையான வருடங்களாக)

Language: TAM Form Language: TAM

Census Data Processing Application (CDAC - GIST, Pune)

File Edit Tools Options Help

CDAC తెలుగు

Page 1 Page 2 Page 3 Page 4 Page 5 Page 6 Page 7 Page 8 Page 9 Page 10

Q 1.1 మొదటి పేరు రవికుమార్

Q 1.2 మధ్య పేరు/తండ్రి పేరు/తల్లి పేరు పుల్లయ్య

Q 1.3 ఇంటి పేరు/కుటుంబం పేరు రాగం

Q 2 గృహయజమానితో సంబంధం క్రొడకు

Q 3 లింగము పురుషుడు

Q 4 గత పుట్టినరోజు వారీకి వయసు (సంవత్సర సంవత్సరములు)

Language: TEL Form Language: TEL

Census Data Processing Application (CDAC - GIST, Pune)

File Edit Tools Options Help

CDAC മലയാളം

Page 1 Page 2 Page 3 Page 4 Page 5 Page 6 Page 7 Page 8 Page 9 Page 10

Q 1.1 ഒന്നാം പേര് രാജീഷ്

Q 1.2 മധ്യ പേര്/ അച്ഛന്റെ

Q 1.3 കുടുംബപ്പേര് നാരായണപിള്ള

Q 2 കുടുംബ നാമസൂചകങ്ങളുടെ ബന്ധం രാജാജി

Q 3 ലിంగം പുരുഷൻ

Q 4 അവസാന അഭിനന്ദനത്തിലെ വയസ്സ് (പൂർത്തിയാക്കിയ വർഷത്തിൽ)

Language: MAL Form Language: MAL

5.0 Data Collection tool for Lite version

- ✦ It is a tool for collecting online Hindi data. Can be used for any language.
- ✦ It is used to collect root characters, aksharas and whole word.
- ✦ User can edit their personal information as well as their online data as and when possible.
- ✦ Every user's information is stored in separate folder with his name and data and

time at which it was collected.

- ✦ User can edit their data in the previous pages by clicking at the previous button.
- ✦ Clear button in every panel allows user to clear all the strokes in that panel. And hence, wrong aksharas or word data can be edited easily.
- ✦ Navigation of pages is easy due to next and previous button.

Screenshots:

Creating New User:

The screenshot shows the 'Go-Write' application window with the 'User Info' dialog box open. The dialog box contains the following fields and options:

- Name: Ash Vape
- Age: 23
- Gender: ☒ Male ☐ Female
- Region: Pune
- Date Of Collection: 15/2/2012
- Education Level: BE/B Tech
- Profession: IT
- Writing Hand: ☒ Right ☐ Left

Buttons at the bottom of the dialog box: OK, Clear, Cancel. The background application window shows a 'New User' button, a 'Virtual Keyboard' button, and 'Previous' and 'Next' navigation buttons.

Figure 5.1 Creating New Users

New user first page:

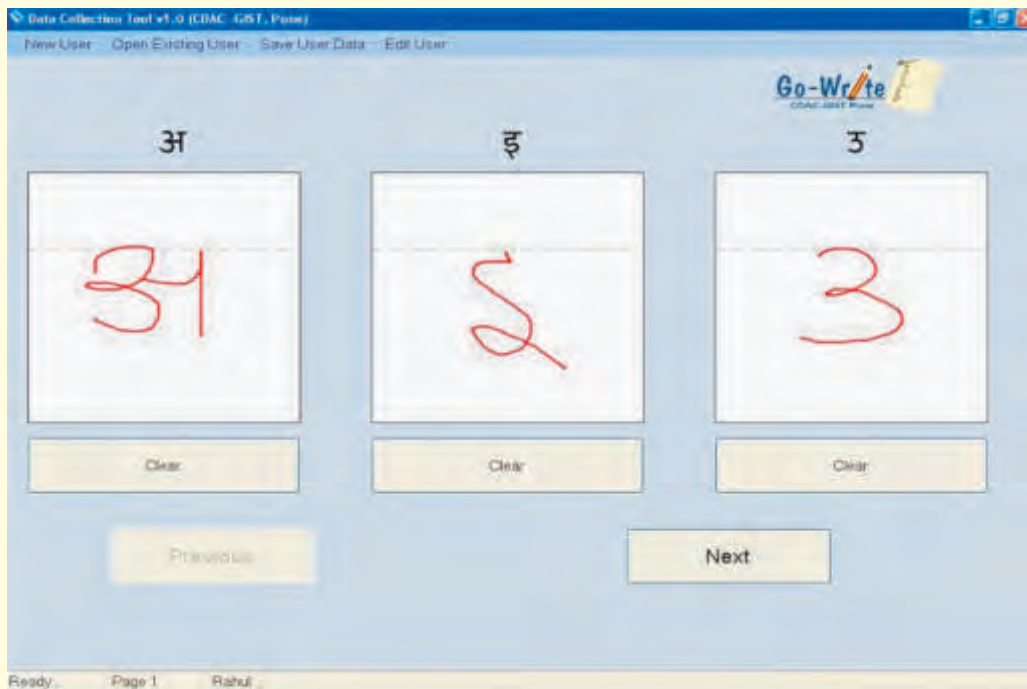


Figure 5.2 Root characters

Opening Existing user Data:

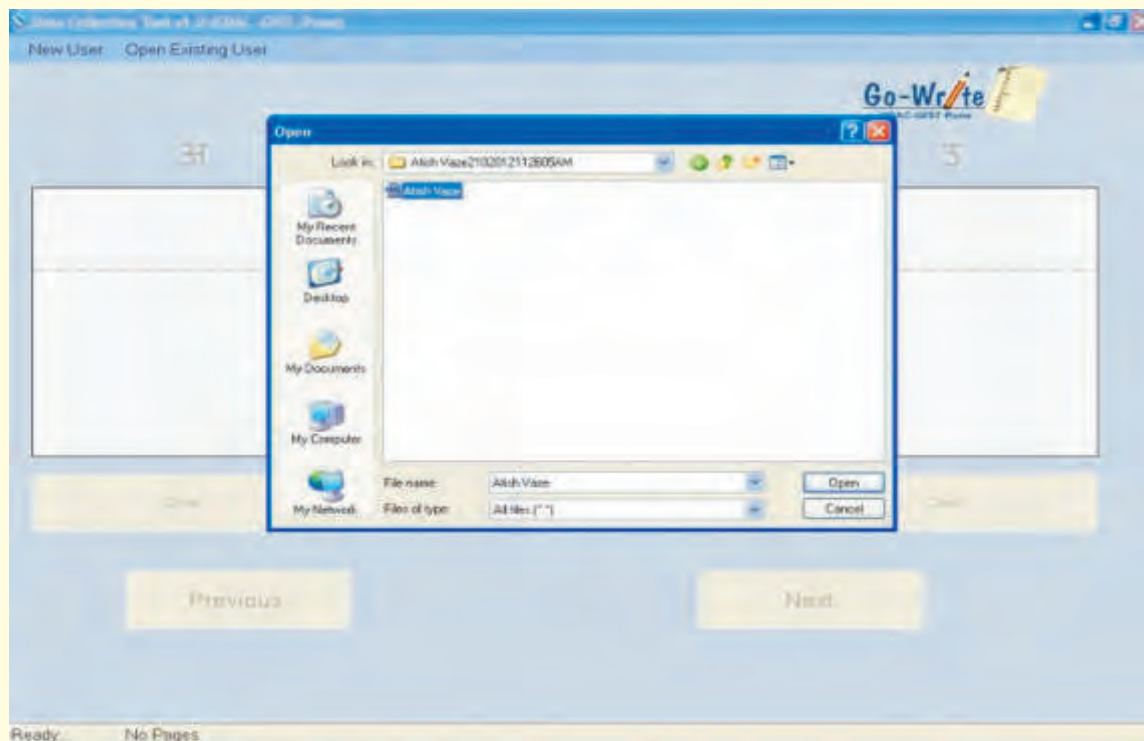


Figure 5.3 Open saved data

Storing of User Data in xml file:

```
<aksharaCount>1</aksharaCount>
<akshara>
  <aksharaNumber>1</aksharaNumber>
  <truthLevel>labeled</truthLevel>
  <labelDesc><desc>A</desc><annotationDetails>
    <codeType>UNICODE</codeType>
    <noOfCodes>1</noOfCodes>
    <codeSequence>2309</codeSequence>
  </annotationDetails>
</labelDesc>
  <sgCount>1</sgCount>
  <strokeGroup>
    <sgNumber>1</sgNumber>
    <strokeCount>2</strokeCount>
    <stroke>
      <strokeNumber>1</strokeNumber>
      <truthLevel>truthed</truthLevel>
      <labelDesc>
        <desc>200</desc>
      </labelDesc>
      <hwTrace><dimension>2</dimension>
      <NoOfPoints>108</NoOfPoints>
      <trace>1905 4471 1905 4471 1905 4498 1905 4498 1905 4524 1905 4577 1905 4604 1931 4656 1984 4762 2011 4762 1
    </hwTrace>
    </stroke>
    <stroke>
      <strokeNumber>2</strokeNumber>
      <truthLevel>truthed</truthLevel>
```

Figure 5.4 XML File for Storing User data

6.0 Some of the Devices Evaluated

The choice of device for generating online handwritten data used for training the engines is very crucial. We have evaluated many devices for suitability for getting the online handwritten data. Some of the major points considered were,

a) ASUS Eee Slate EP121 Tablet :

- Ease of writing during long writes
- Availability of SDK
- Availability of stylus/pen

These devices were also looked for possible field trials. Following are some of the devices that we found suitable for this project.



Figure 6.1 ASUS Tablet

Specification:

Operating System	Genuine Windows® 7 Home Premium
CPU	Intel dual Core i5
Display	12.1" LED Backlight Capacitive Screen
Storage	32GB / 64GB

Link: http://www.asus.com/Eee/Eee_Pad/Eee_Slate_EP121/#overview

b) HTC Flyer Tablet :



Figure 6.2 HTC Flyer Tablet

Specification:

Operating System	Android
CPU	1.5 GHz
Display	7" LCD capacitive touchscreen
Storage	32GB

Link: <http://www.htc.com/in/tablets/htc-flyer/#specs>

c) Samsung Galaxy Note :



Figure 6.3 Samsung Galaxy Note Smartphone

Specification:

Operating System	Android 2.3(Gingerbread)
CPU	Dual Core 1.4 GHz
Display	5.3" LED capacitive touchscreen
Storage	32GB

Table 5.3 Specification for Samsung Galaxy Note Smartphone

Link: <http://www.samsung.com/in/galaxynote/>

d) Dell Latitude XT3 Tablet PC :



Figure 6.4 Dell Tablet PC

Specification:

Operating System	Windows 7
CPU	
Display	13.3" LED capacitive touchscreen

Table 5.4 Dell Tablet PC

Link: <http://www.dell.com/in/business/p/latitude-xt3/pd>

e) Hi-Tech tablet Note-Taker :



Figure 6.5 Hi-Tech tablet Note-Taker

Specification:

Title	Specification
Dimension	?
Work Area	Upto A4
Resolution	100 dpi
Standards	Meet CE and BSMI radiation standard
Operating System	Win 2000/XP

Link: http://www.hitech-in.com/pcnote_taker.htm

We recommend Dell XT3 tablet PC for data collection required for development of engines, while as Hi-Tech's Note taker for on-field uses. The Note taker is cost effective and retains the hard copy of the form. Dell XT3 gives the sturdiness required for exhaustive data collection work.

7.0 New XML standard

The new XML standard incorporates the sub-stroke, sentence level annotation information.

Following new tags are introduced in revised version of OHWR XML schema.

1. Text-block

2. Paragraph

3. Sub-stroke

Also provide attributes to store sentence level information. In revised XML schema we divide page level data into text-block and paragraph level to deal with text area and non-text area in handwritten documents. Refer following figure 7.1

Sometimes in case of Bengali handwritten word single stroke may be part of more than one akshara so it's necessary to break stroke into sub-strokes (figure 7.2).

```
<annotationDef>
  <pointOfOrigin>1</pointOfOrigin>
  <document encodingType="UTF-8" traceDimension="2" pageCount="1">
    <page pageNumber="1" pageId="HN_CDACP_USER_001_1" textBlockCount="1">
      <textBlock textBlockNumber="1" truthLevel="labeled" paragraphCount="1" noOfCorners="4">
        <polygon>0 0 8500 0 8500 11000 0 11000</polygon>
        <paragraph paragraphNumber="1" truthLevel="labeled" lineCount="1" sentenceCount="1">
          <line lineNumber="1" truthLevel="labeled" wordCount="2">
```

Figure 7.1 Revised OHWR XML schema

```
<stroke strokeNumber="1" truthLevel="labeled" subStrokeCount="2">
  <labelDesc>
    <desc>19</desc>
  </labelDesc>
  <subStroke subStrokeNumber="1" truthLevel="labeled" subStrokeID="1">
    <labelDesc>
      <desc>0</desc>
    </labelDesc>
    <hwTrace NumberOfPoints="42">
      <trace>1708 9934 1708 9935 1707 9935 1705 9937 1704 9938 1703 9938 1702 9938 1703 9938 1704 9938 1705 9938 1707
9938 1710 9938 1714 9938 1720 9938 1723 9938 1729 9938 1735 9938 1744 9938 1751 9940 1760 9941 1767 9943 1774 9943
1781 9943 1788 9943 1796 9943 1802 9943 1809 9945 1813 9947 1816 9948 1818 9949 1819 9950 1820 9951 1823 9953 1824
9955 1826 9955 1825 9954 1826 9953 1826 9953 1826 9952 1824 9952 1824 9953 1824 9953</trace>
    </hwTrace>
```

```

</subStroke>
<subStroke subStrokeNumber="2" truthLevel="labeled" subStrokeID="2">
  <labelDesc>
    <desc>0</desc>
  </labelDesc>
  <hwTrace NumberOfPoints="60">
    <trace>1754 10062 1753 10062 1751 10062 1750 10062 1751 10062 1751 10062 1751 10062 1751 10063 1749 10063 1748
    10063 1745 10063 1743 10063 1742 10063 1741 10064 1738 10064 1737 10064 1736 10064 1734 10064 1733 10064 1733
    10065 1731 10065 1731 10066 1731 10066 1733 10066 1736 10066 1740 10066 1747 10066 1753 10066 1758 10066 1765
    10066 1777 10066 1789 10066 1802 10066 1818 10066 1831 10066 1844 10066 1855 10066 1867 10066 1878 10066 1887
    10066 1896 10066 1903 10066 1909 10066 1914 10066 1918 10066 1921 10066 1922 10066 1924 10066 1925 10066 1926
    10066 1926 10067 1926 10068 1926 10069 1926 10071 1926 10072 1926 10073 1925 10073 1923 10074 1921 10074 1921 10074
    </trace>
  </hwTrace>
</subStroke>
</stroke>

```

Figure 7.2 Sub-stroke level data

Most of tags in previous version of OHWR XML schema which represent property of parent tags are converted to attributes of

respected tag in revised version of OHWR XML schema. Refer figure 7.3 and 7.4.

```

<document>
  <originCode>1</originCode>
  <noOfPages>24</noOfPages>
  <page>
    <pageNo>1</pageNo>
    <pageId>HN_CDACP_USER_001_1</pageId>
    <lineCount>12</lineCount>
    <line>
      <lineNumber>1</lineNumber>
      <truthLevel>labeled</truthLevel>
      <labelDesc>
        <desc>A + AA + I + II + U + UU</desc>
        <annotationDetails>
          <codeType>UNICODE</codeType>
          <noOfCodes>11</noOfCodes>
          <codeSequence>2309 32 2310 32 2311 32 2312 32 2313 32
          2314</codeSequence>
        </annotationDetails>
      </labelDesc>
      <wordCount>6</wordCount>
      <word>
        <wordNumber>1</wordNumber>
        <truthLevel>labeled</truthLevel>

```

Figure 7.3 Previous version of OHWR XML schema

Figure 7.4 Revised version of OHWR XML schema

9.0 References

1. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998.
2. Hailong Liu; Xiaoqing Ding; , "Handwritten character recognition using gradient feature and quadratic classifier with multiple discrimination schemes," Proceedings. Eighth International Conference on Document Analysis and Recognition ICDAR, 2005, Vol 1, pp 19-23.
3. Speech and language processing: An Introduction to Speech recognition, computational Linguistics and natural language processing. Daniel Jurafsky & James H. Martin
4. http://en.wikipedia.org/wiki/Language_model
5. <http://www-speech.sri.com/projects/srilm/>

