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 on if 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 cater 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:

```
कहें दुनिा कहीं कहीं कहीं कहीं है वहीं
कहीं गैरियों लोगों लोगों लें बनाकरी भ्रम लगाईँ लगाईँ
कहें वाद कर वाद कर वाद कर वाद कर वाद कर वाद कर वाद कर वाद कर
कहें तक न चाहिए अंक तक न चाहिए अंक तक न चाहिए अंक तक न चाहिए अंक तक न चाहिए अंक तक न चाहिए अंक तक न
कहें आगे चल आगे चल आगे चल आगे चल आगे चल आगे चल आगे चल आगे
कहें अच्छे अच्छे जीवन भरे अच्छे अच्छे जीवन भरे अच्छे अच्छे जीवन भरे अच्छे अच्छे जीवन भरे अच्छे अच्छे जीवन भरे अच्छे अच्छे जीवन भरे
कहें अंतत्त्व कहें अंतत्त्व कहें अंतत्त्व कहें अंतत्त्व कहें अंतत्त्व कहें अंतत्त्व कहें अंतत्त्व कहें अंतत्त्व कहें अंतत्त्व कहें
कहें अंतत्त्व कहें अंतत्त्व कहें अंतत्त्व कहें अंतत्त्व कहें अंतत्त्व कहें अंतत्त्व कहें अंतत्त्व कहें अंतत्त्व कहें अंतत्त्व कहें
```

#### 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 layer.
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 Mj 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
   \[ \partial X = X - X_{\text{ideal}} \]

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.
   \[ \partial Y = \log(1 + \exp(-Y)) \ast \partial X \]

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:
   \[ w_{\text{new}} = w_{\text{old}} - \alpha \ast \partial w \]
   where \( \alpha \) is the learning rate, and a very important design parameter.

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

\[ \nabla w_{ij} = -\varepsilon \frac{\partial C}{\partial w_{ij}} \]

Now becomes:

\[ \nabla w_{ij} = -\varepsilon \frac{\partial C}{\partial w_{ij}} \]

The Second Order methods, with the above given Pseudo-Newton Step, converge 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](image)
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.

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

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trained Data (22758 examples)</td>
<td>99.6%</td>
</tr>
<tr>
<td>Test Data (12134 examples)</td>
<td>96%</td>
</tr>
</tbody>
</table>

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 samples so as to speed up the training.

**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.

**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 shirorekha. Character segmentation is hybrid (online + offline) approach for segmenting symbols from handwritten word.

Shirorekha and Matras Seperation are first step. Shirorekha 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 Shirorekha and matras. Then Root Aksharas are separated using geometrical information and rule based on spatial information.

Shirorekha and Matras Seperation are first step. Shirorekha 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 Shirorekha and matras. Then Root Aksharas are separated using geometrical information and rule based on spatial information.

**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, conjucts, Shirorekha (baseline). Besides, matras can appear in any region.

For the handwritten word, segmentation is more difficult than printed word (OCR). Handwritten data is more complex (as shown in Fig.2.10) because Akshras 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.

**Training:**

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

Online handwritten data contains series of
strokes which contains array X, Y points from pen up to pen down. Group of Strokes can form a Symbol. Symbol contains root (main) character, Akshara or matras. There are different ways (combination of strokes) to write a following symbol as shown in Fig.2.11.

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 ∈ {a, 2...176} and there are 150 Symbols (Smb).

\[ Smb \in \{स, अ, ए, इ, आ, ऑ, ओ, औ,...\} \]

Language model is widely used in NLP and speech recognition. In Language model probabilities are calculated for n-grams from corpus. Same thing we applied for strokes sequence. We have calculated probabilities for sequence of one stroke, two strokes and three strokes for all symbols.

\[ P(S_t), P(S_t|S_{t-1}), P(S_t|S_{t-1}, S_{t-2}) \]

Data is too sparse for computing probabilities. Few strokes are appeared in a symbol. Most of Stroke counts are zero. We smooth these probabilities using Good-Turing Smoothing technique. From Annotated dataset of 50 writers we are having different types stroke sequence data with labeled 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.

**Recognition:**

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.

\[
Smb_1 = \underset{Smb}{\text{argmax}} \ P(S_t)
\]

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

\[
Smb_3 = \underset{Smb}{\text{argmax}} \ P(S_t | S_{t-1}, S_{t-2})
\]

Tree is formed recursively with all linear sequence of strokes with length 1, 2 and 3 shown in Fig.2.12.

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 (Smb) 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.
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.

Future Work:

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.

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.

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.

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

4.0 Census Data Collection Application (Tablet-PC Edition)
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.
Following are the screens of the Census application for Tablet-PC.
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:

Figure 5.1 Creating New Users
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.

New user first page:

Figure 5.2 Root characters

Opening Existing user Data:

Figure 5.3 Open saved 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:

![Figure 6.1 ASUS Tablet](http://www.asus.com/Eee/Eee_Pad/Eee_Slate_EP121/#overview)

**Specification:**

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Genuine Windows® 7 Home Premium</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Intel dual Core i5</td>
</tr>
<tr>
<td>Display</td>
<td>12.1” LED Backlight Capacitive Screen</td>
</tr>
<tr>
<td>Storage</td>
<td>32GB / 64GB</td>
</tr>
</tbody>
</table>

**Link:** [http://www.asus.com/Eee/Eee_Pad/Eee_Slate_EP121/#overview](http://www.asus.com/Eee/Eee_Pad/Eee_Slate_EP121/#overview)

- Ease of writing during longe 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.

- HTC Flyer Tablet:

![Figure 6.2 HTC Flyer Tablet](http://www.htc.com/in/tablets/htc-flyer/#specs)

**Specification:**

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Android</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>1.5 GHz</td>
</tr>
<tr>
<td>Display</td>
<td>7” LCD capacitive touchscreen</td>
</tr>
<tr>
<td>Storage</td>
<td>32GB</td>
</tr>
</tbody>
</table>

**Link:** [http://www.htc.com/in/tablets/htc-flyer/#specs](http://www.htc.com/in/tablets/htc-flyer/#specs)

- Samsung Galaxy Note:

![Figure 6.3 Samsung Galaxy Note Smartphone](http://www.samsung.com/in/galaxynote/)

**Specification:**

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Android 2.3(Gingerbread)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Dual Core 1.4 GHz</td>
</tr>
<tr>
<td>Display</td>
<td>5.3 “ LED capacitive touchscreen</td>
</tr>
<tr>
<td>Storage</td>
<td>32GB</td>
</tr>
</tbody>
</table>

**Link:** [http://www.samsung.com/in/galaxynote/](http://www.samsung.com/in/galaxynote/)
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,

- 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.

b) ASUS Eee Slate EP121 Tablet:

![Figure 6.1 ASUS Tablet](http://www.asus.com/Eee/Eee_Pad/Eee_Slate_EP121/#overview)

**Specification:**

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Genuine Windows® 7 Home Premium</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Intel dual Core i5</td>
</tr>
<tr>
<td>Display</td>
<td>12.1” LED Backlight Capacitive Screen</td>
</tr>
<tr>
<td>Storage</td>
<td>32GB / 64GB</td>
</tr>
</tbody>
</table>

**Link:** [http://www.asus.com/Eee/Eee_Pad/Eee_Slate_EP121/#overview](http://www.asus.com/Eee/Eee_Pad/Eee_Slate_EP121/#overview)

b) HTC Flyer Tablet:

![Figure 6.2 HTC Flyer Tablet](http://www.htc.com/in/tablets/htc-flyer/#specs)

**Specification:**

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Android</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>1.5 GHz</td>
</tr>
<tr>
<td>Display</td>
<td>7” LCD capacitive touchscreen</td>
</tr>
<tr>
<td>Storage</td>
<td>32GB</td>
</tr>
</tbody>
</table>

**Link:** [http://www.htc.com/in/tablets/htc-flyer/#specs](http://www.htc.com/in/tablets/htc-flyer/#specs)

c) Samsung Galaxy Note:

![Figure 6.3 Samsung Galaxy Note Smartphone](http://www.samsung.com/in/galaxynote/)

**Specification:**

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Android 2.3(Gingerbread)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Dual Core 1.4 GHz</td>
</tr>
<tr>
<td>Display</td>
<td>5.3” LED capacitive touchscreen</td>
</tr>
<tr>
<td>Storage</td>
<td>32GB</td>
</tr>
</tbody>
</table>

**Table 5.3 Specification for Samsung Galaxy Note Smartphone**

**Link:** [http://www.samsung.com/in/galaxynote/](http://www.samsung.com/in/galaxynote/)
d) Dell Latitude XT3 Tablet PC:

![Dell Tablet PC](image)

**Specification:**

<table>
<thead>
<tr>
<th>Title</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>Windows 7</td>
</tr>
<tr>
<td>CPU</td>
<td></td>
</tr>
<tr>
<td>Display</td>
<td>13.3” LED capacitive touchscreen</td>
</tr>
</tbody>
</table>

Table 5.4 Dell Tablet PC

[Link](http://www.dell.com/in/business/p/latitude-xt3/)  

e) Hi-Tech tablet Note-Taker:

![Hi-Tech tablet Note-Taker](image)

**Specification:**

<table>
<thead>
<tr>
<th>Title</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension</td>
<td>?</td>
</tr>
<tr>
<td>Work Area</td>
<td>Upto A4</td>
</tr>
<tr>
<td>Resolution</td>
<td>100 dpi</td>
</tr>
<tr>
<td>Standards</td>
<td>Meet CE and BSMI radiation standard</td>
</tr>
<tr>
<td>Operating System</td>
<td>Win 2000/XP</td>
</tr>
</tbody>
</table>

[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).

Figure 7.1 Revised OHWR XML schema
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.
8.0 New Application Interfaces (API)

The new interfaces are added to the document, it now includes API for text block recognition. The new interfaces are kept backward compatible.

**Method Summary:** This function will call the engine and pass the bag-of-strokes corresponding to text block. There will not be any segmentation from the application side. The engine will return the Unicode strings in *ptrTextBlockUniInfo.

```
fn_RecognizeTextBlock()
```

**Input:**
1. pStructStrokeInfo :- Filled FILE_STROKEINFO structure with word Information.
2. nStrokeCount:- Total number of strokes
3. topN :- This parameter is passed by application to engine. It asks the engine to return top N outputs in *ptrTextBlockUniInfo. By default the value passed is zero. It can be used by those engines that support this feature.
4. ptrTextBlockUniInfo:- This is a pointer to astructure GTEXT_BLOCK_UNI_INFO. The structure is defined in Annexure-1.

The engine returns this structured with line, word, stroke information and the Unicode characters belonging to each word.

**Returns:**
1) 0 for success
2) Negative Integers for Errors codes

* Errors are engine errors and have to be listed to application so that they can be displayed in message box.

**Prototype:**

```
int fn_RecognizeTextBlock (const FILE_STROKEINFO* pStructStrokeInfo, / int nStrokeCount , int topN, / GTEXT_BLOCK_UNI_INFO * ptrTextBlockUniInfo);
```
9.0 References


3. Speech and language processing: An Introduction to Speech recognition, computational Linguistics and natural language processing. Daniel Jurafsy & James H. Martin


***