Research Report on Bangla OCR Training and Testing Methods

Md. Abul Hasnat
BRAC University, Dhaka, Bangladesh.
hasnat@bracu.ac.bd

Abstract

In this paper we present the training and recognition mechanism of a Hidden Markov Model (HMM) based multi-font Optical Character Recognition (OCR) system for Bengali character. In our approach, the central idea is to separate the HMM model for each segmented character or word. The system uses HTK toolkit for data preparation, model training and recognition. The Features of each trained character are calculated by applying the Discrete Cosine Transform (DCT) to each pixel value of the character image where the image is divided into several frames according to its size. The extracted features of each frame are used as discrete probability distributions which will be given as input parameters to each HMM model. In the case of recognition, a model for each separated character or word is built up using the same approach. This model is given to the HTK toolkit to perform the recognition using the Viterbi Decoding method. The experimental results show significant performance over models using neural network based training and recognition systems.

1. Introduction

Hidden Markov Models (HMM) have been used in a wide range of NLP applications, ranging from continuous speech recognition to handwriting recognition [1,2]. We use the similar idea of applying HMM for speech recognition into our training and recognition approach of OCR. The core idea in our approach is to create model for each individual segmented character or word. We use HTK Toolkit for implementing our application which has been used after completing the preprocessing and feature extraction steps. After performing image processing task we will specify the number of states and the features. Taking this as input parameter HTK Toolkit will estimate the allowable transitions and probability distribution among the states and create the appropriate model for the given character or word.

Preprocessing includes binary image conversion, noise removing from image, skew correction, segmentation which includes line separation, word separation and character separation. In case of complexity in character separation segmentation is performed at word level. The segmented portion is then converted into image array which contains binary values for each pixel. Note that in our approach the segmented character is obtained by considering the image boundary not the connected component approach.

The next step is to divide the image array into certain number of frames which will be calculated according to the width of the image array. These frame will be consider as the number of states for the HMM model of the segmented image.

Feature extraction is performed after preprocessing steps and framing are done successfully. This methodology includes Discrete Cosine Transform (DCT) calculation on each individual pixel of each frame and modifies the pixel value in each frame accordingly. These newly calculated pixels value will be considered as discrete probability distributions which will be given as input parameter to each HMM model.

The procedure of creating a particular model is done by HTK Toolkit from the number of samples that is provided as training input and then save this model description into appropriate files. Based on the number of samples the model parameters (means and variance) will be estimated by internal command of HTK Toolkit.

But for recognition a temporary HMM model is build from the word or character image using the same procedure described above. Then the recognition is performed using Viterbi decoding by HTK Toolkit.

The post processing includes taking the appropriate Unicode characters from the model database against each model name which is determined by the recognition tools of HTK.

Then these characters are stored and written into file sequentially.

The flow chart for training and recognition is shown in Figure 1 that clearly visualizes the actual procedure of training and recognition.

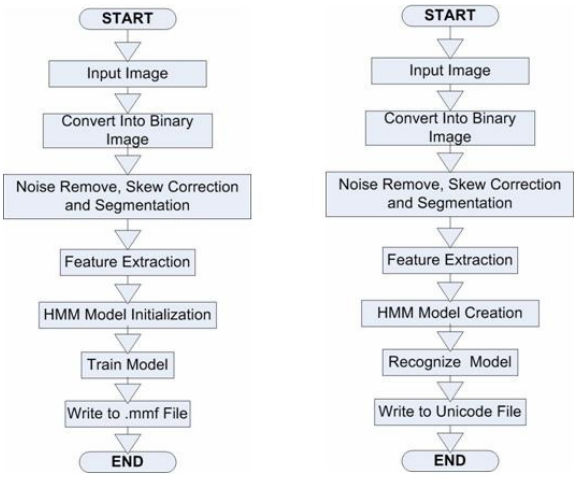

Figure 1: Flow chart of training and recognition procedure

2. Overview

The entire procedure of recognition and training can be break down into two parts based on HTK Toolkit point of view. The first part (section 2.1) consists of preprocessing up to Feature calculation which is almost similar for both the operation of training and recognition. The second part (section 2.2) consists of HTK Toolkit operation which is different for training and recognition process. In other word the first part can be renamed as the "Image data input for HTK Toolkit". Here I will discuss the first part starting from segmented image to feature calculation and the second part separately for both training and recognition.

2.1. Segmented image to feature calculation

Here I assume that I have already got the segmented image that can be either a character or a word and the image is already converted to binary image. Let take a segmented character and a segmented word which is shown in Figure 2.

(a) (b)

Figure 2: (a) Segmented character 'soreo'; (b) Segmented word 'tumi'

2.1.1. Frame calculation. Now from these images number of frame will be calculated. In our approach we choose the frame width to be 8 and the frame height to be 90. The frame width and height is chosen according to our statistical analysis. Based on the frame width and height we divide the segmented image into several frames. The size of mean and variance vector is also determined from the frame width and height. For example the number of frame of the segmented character soreo is 3 and segmented word tumi has 6 frames. Number of frame is most important because it determines the number of states of the HMM model. So we can say that the number of states for hmm model soreo is 3 and tumi has 6 states. The above discussion is illustrated in Figure 3 and Figure 4.

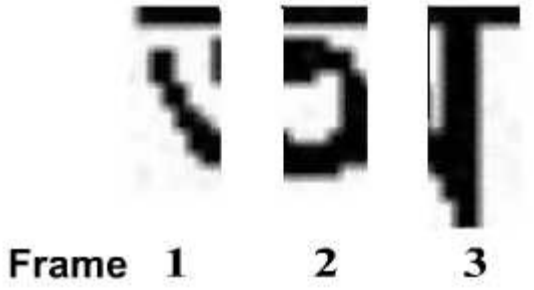

Figure 3: segmented character "soreo" with 3 frames

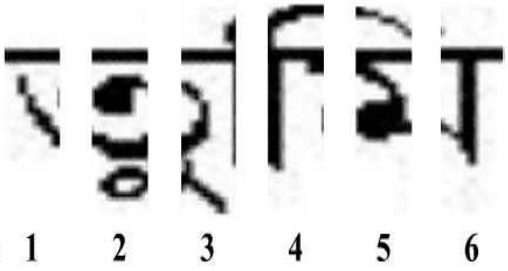

Figure 4: segmented word "tumi" with 3 frames

2.1.2. Feature calculation. In this phase each frame is taken separately and then the feature calculation is performed on each individual pixels of the frame by applying Discrete Cosine Transform (DCT).

2.1.3. Discrete Cosine Transform (DCT). The Discrete Cosine Transform (DCT) converts a continuous signal into its elementary frequency

components, and as such, closely related to the Discrete Fourier Transform (DFT). The DCT of an image can represent it as a sum of sinusoids of varying magnitude and frequencies. Because the DCT of an image captures the most visually significant information in just a few coefficients, it is widely used in image algorithms such as compression. DCT of an image is given by the following:

$$B_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{mn} \cos\frac{\pi(2m+1)p}{2M} \cos\frac{\pi(2n+1)q}{2N}, \quad \begin{matrix} 0 \le p \le M-1 \\ 0 \le q \le N-1 \end{matrix}$$

$$\alpha_p = \begin{cases} 1/\sqrt{M}, & p = 0 \\ \sqrt{2/M}, & 1 \le p \le M-1 \end{cases} \qquad \alpha_q = \begin{cases} 1/\sqrt{N}, & q = 0 \\ \sqrt{2/N}, & 1 \le q \le N-1 \end{cases}$$

where A and B are the input and output images of M rows and N columns respectively. The DCT is a linearly separable transformation, so computing a two-dimensional DCT can be done with one-dimensional DCTs along each dimension. Figure 5 shows the computation steps of a two-dimensional DCT.

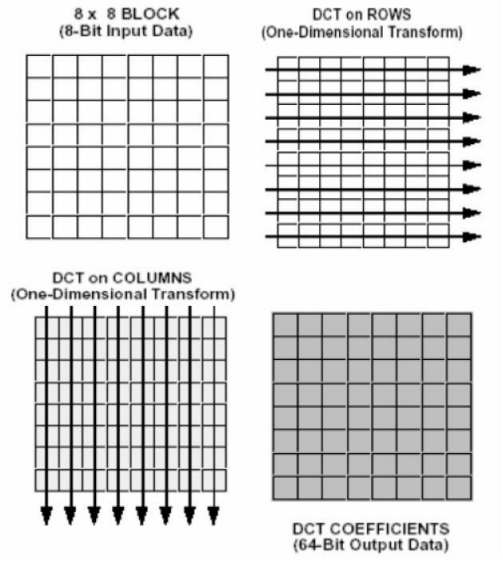

Figure 5: Two-dimensional DCT transform

After applying DCT on the pixels values of each frame the updated information is stored into a file which will be given as hmm model input for the particular character or word that is considered to be trained.

Up to this stage the image processing methodology is exactly same for both the operation of training and recognition. The only difference is that, training mechanism uses a certain number of samples for modeling a particular character or word that is used for estimating model parameters in HTK Toolkit but recognition mechanism create model only for the particular image character or word to be recognized.

So at this stage we can create HMM model for the character "soreo" and word "tumi" which is shown in Figure 6 and Figure 7.

These models clearly prove the description of the framing and feature extraction methodology described above. We can see from these models that the segmented character "soreo" has 3 states without the start and end state that is common to each model. Similarly the segmented word has 6 states without the start and end states. Now the HMM model components (number of states and calculated feature values) for the segmented character and word constructed in this stage is ready to be given as HTK Toolkit input for initializing the HMM model or performing the recognition task.

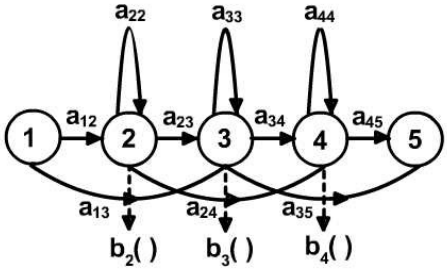

Figure 6: HMM model for character "soreo"

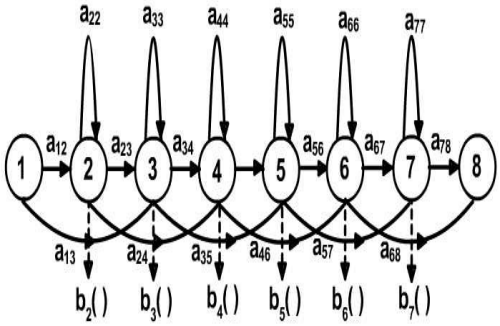

Figure 7: HMM model for word "tumi"

2.2. HTK toolkit operations for training and recognition

2.2.1. The HTK toolkit. The Hidden Markov Model Toolkit (HTK) was initially designed for continuous speech recognition technology development; in recent years, it has been used for everything from speech

synthesis and character recognition to DNA sequencing. See [13] for more details on HTK.

Parts of HMM modeling using HTK Toolkit are divided into four phases:

1. Data Preparation
2. Model Training
3. Pattern Recognition
4. Model Analysis

2.2.2 Data preparation. In the Data Preparation stage first we will define a word network of word-to-word transition using HTK's Standard Lattice Format (SLF). HTK also provides a grammar definition format, and an associated HParse tool, that can be used to build this word network automatically. We write the grammar definition in a file called gram.txt.

For our application the grammar is as follows:

$WORD = h0900|h0901|h0902;([$WORD])

Here in the grammar the model name of the word or character that will be trained is specified as **h####**. This is the convention for writing model name in our application and the same convention is used everywhere in this application.

We now create a sorted list of the character or sequence of characters to be trained, and save that in a file called dict.txt. The dictionary file syntax is given below:

h0900 [h0900] h0900
h0901 [h0901] h0901
h0902 [h0902] h0902

For each line, the initial string specifies the model, the second string in brackets specifies the string to output, and the final string specifies the output transcription against the model name. Now we will use the tool HSGen to generate the prompts for test sentences. Note that the data preparation stage is required only for recognition purpose. It has absolutely no usage for the training purpose.

2.2.3. Model training. The first task is to define a prototype for the HMM model to be trained, which includes the model topology, and transition and output distribution parameters. This task will of course depend on the number of states and the extracted feature of each character or word. In our application, the observation state is finite for each frame (8 x 90 = 720 states), so we use discrete distributions. The prototype HMM definition of the model "soreo" is given in Figure 8.

```
~h "h000"
<BeginHMM>
<VecSize> 720 <USER>
<NumStates> 5
<State> 2
<Mean> 720
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 .....
<Variance> 720
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 .....
<State> 3
<Mean> 720
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 .....
<Variance> 720
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 .....
<State> 4
<Mean> 720
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 .....
<Variance> 720
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 .....
<TransP> 5
0.0 0.5 0.5 0.0 0.0
0.0 0.4 0.4 0.2 0.0
0.0 0.0 0.6 0.4 0.0
0.0 0.0 0.0 0.7 0.3
0.0 0.0 0.0 0.0 0.0
<EndHMM>
```

Figure 8: HMM prototype definition for model "soreo"

From the model definition above, we see that the number of states is specified as 5 and each individual state has mean and variance vectors. The mean vectors are initialized with value 0.0 and the variance vector is initialized with value 1.0. At the end of the definition we see the transition probability matrix where the dimension of the matrix is 5 by 5.

The HMM model parameters contain the probability distribution or estimation of each model, which we now have to initialize. By far the most prevalent probability density function is the Gaussian probability function that consists of means and variances and this density function are used to define the model parameters in HTK. We use HTK's HInit tool to provide the initial estimates using a set of observation sequences, producing a complete HMM model definition with estimated parameters. The entire process of initialization is visualized in Figure 9.

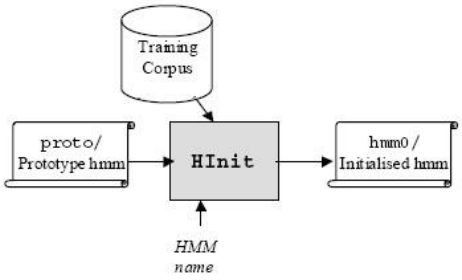

Figure 9: Initialization process of an HMM model

After the initialization process is completed the HMM model is written into .mmf file that contains all the trained models and is used in recognition.

2.2.4. Pattern recognition. Comparative to the training, recognition is much simpler. To complete this task we have to create a HMM model of the character or word image that will build up from the first part (section 2.1) of our system. Then this model will match with all the HMM models and the most likely model will be given as output.

To perform this task using HTK we have to invoke the recognition tool HVite, which is a general purpose Viterbi word rcognizer. HVite uses the word network describing the allowable word sequence built from the task grammar, the dictionary that defines each character or word, the entire list of HMMs and the .mmf file where the description of each HMM model is written. HVite recognizes an HMM model by matching it against a network of models, and then writes out the transcription for the recognized model into a Master Label File with .mlf extension. The entire process of recognition is visualized in Figure 10.

After the recognition process is completed the model name is read from the Master Label File (.mmf) and the associated Unicode character for the recognized model is written to the output file. An example of Master Labeled File is given below:

!MLF!#
"C:/htk/data/train/user/1.rec"
0 3000 h0900 2976.731201

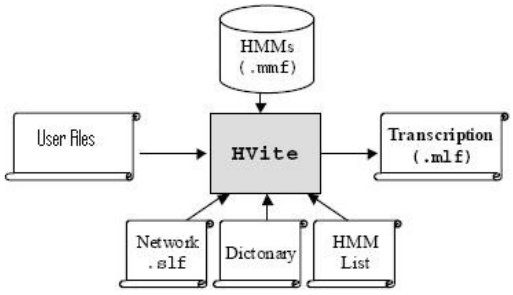

Figure 10: Recognition process of a HMM model

3. Required HTK commands

3.1. HParse

The HParse program takes a grammar definition file (based on extended Backus-Naur Form or EBNF) of the word network, and generates the word level lattice files in HTK's SLF format. HVite then uses the lattice file for pattern recognition.

HParse is invoked as following:
HParse [options] syntaxFile latFile

For example: HParse -A -D -T 1 gram.txt net.slf

3.2. HSGen

The HSGen randomly generates sentences from the regular language L(G), where the regular grammar G is implicitly defined in the word network. HSGen reads the lattice file, and writes out the sentences, one per line, to the standard output.

HSGen is invoked as following:
HSGen [options] wdnet dictfile

For example: HSGen -A -D -n 10 -s net.slf dict.txt

3.3. HInit

The HInit tool used to train the HMM model, by providing the initial estimates for the model parameters using a set of observation sequences. It does so by repeatedly applying Viterbi alignment to to segment the training observations and then recomputing the parameters, and produces a complete HMM model definition with estimated parameters

HInit is invoked as following:
HInit [options] hmm trainFiles ...

For example: HInit -A -D –T 1 -S trainlist.txt -M model/hmm0 \ -H model/proto/hmmfile -l label -L label_dir nameofhmm

3.4. HVite

HVite is used for the pattern recognition. It recognizes a model by matching it against a network of

models, and writes out the transcription of the recognized model into a master label file.

HVite is invoked as following:
HVite [options] dictFile hmmList testFiles ...

For Example: HVite -A -D -T 1 -H hmmsdef.mmf -i reco.mlf -w net.slf \dict.txt hmmlist.txt input.mfcc

4. Conclusion

We present the training and recognition mechanism of the Hidden Markov Model (HMM) based Optical Character Recognizer (OCR) for Bengali character. The system is mainly divided into two parts; the first one is the preprocessing step, leading up to the the feature extraction and the other part consists of the HTK Toolkit operations that are used to initialize the HMM model from a certain number of samples and to recognize a particular model. The current system has been trained on a single font using a single 25-point font size. We are now beginning to create the training set for the multi-font and multi-fontsize training and recognition. Since the number of trained models has has strong impact on the performance of the models, we are focusing on creating a larger training set.

5. References

[1] Z. Lu, I. Bazzi, A. Kornai, J. Makhoul, P. Natarajan and R. Schwartz, "A Robust, Language-Independent OCR System", Proc. *SPIE Vol. 3584, 27th AIPR Workshop: Advances in Computer-Assisted Recognition*, 1999, pp. 96-104.

[2] W. Wang, A. Brakensiek, A. Kosmala and G. Rigoll, "HMM Based High accuracy off-line cursive handwriting recognition by a baseline detection error tolerant feature extraction approach", *7th Int. Workshop on Frontiers in Handwriting Recognition (IWFHR)*, 2000.

[3] U. Pal, B.B. Chaudhuri, "OCR in Bangla: an Indo-Bangladeshi Language", *Pattern Recognition, Vol. 2 - Conference B: Computer Vision & Image Processing, Proceedings of the 12th IAPR International*, 1994.

[4] B.B. Chaudhuri, U. Pal, "An OCR System to Read Two Indian Language Scripts: Bangla and Devnagari (Hindi)", IEEE Computer Society, 1997.

[5] A. Bishnu, B.B. Chaudhuari, "Segmentation of Bangla Handwritten Text into Characters by Recursive Contour Following", *IEEE Computer Society*, 1999.

[6] U. Garain, B.B. Chaudhuri, "Segmentation of touching characters in Printed Devnagari and Bangla Scripts using Fuzzy Multifactorial Analysis", *IEEE Computer Society*, 2001.

[7] J.U. Mahmud, M.F. Raihan and C.M. Rahman, "A Complete OCR System for Continuous Bangla Characters", *IEEE TENCON-2003: Proceedings of the Conference on Convergent Technologies for the Asia Pacific*, 2003.

[8] A.A. Chowdhury, E. Ahmed, S. Ahmed, S. Hossain and C.M. Rahman, "Optical Character Recognition of Bangla Characters using neural network: A better approach", *2nd International Conference on Electrical Engineering (ICEE)*, Khulna, Bangladesh, 2002.

[9] A. Kundu, P. Bahl and Y. Yang, "Recognition of Handwritten Word: First and Second Order Hidden Markov Model Based Approach", *Journal of the Pattern Recognition Society*, Pergamon Press, Vol. 22, No. 3, 1989, pp. 283-297

[9] W. Wang, A. Brakensiek, A. Kosmala, and G. Rigoll. "HMM based high accuracy off-line cursive handwriting recognition by a baseline detection error tolerant feature extraction approach", In *Proc. Int. Workshop on Frontiers in Handwriting Recognition*, Amsterdam, 2000, pp. 209-218.

[10] J. Doménech, A.H. Toselli, A. Juan, E. Vidal, and F. Casacuberta, "An off-line HTK-based OCR system for isolated handwritten lowercase letters", In *Proc. of the IX Spanish Symposium on Pattern Recognition and Image Analysis, volume II*, Benicàssim, Spain, May 2001, pp. 49-54.

[11] S.A. Al-Qahtani and M.S. Khorsheed, "A HTK-Based System to Recognise Arabic Script", in *Proc. 4th IASTED International Conference on Visualization, Imaging, and Image Processing*, Marbella, Spain, ACTA Press, 2004.

[12] S.A. Al-Qahtani and M.S. Khorsheed, "An Omni-font HTK-Based Arabic Recognition System", in *Proc. 8th IASTED International*

Conference on Artificial Intelligence and Soft Computing, Marbella Spain, ACTA Press, 2004.

[13] The HTK Book (for HTK Version 3.3) available at http://htk.eng.cam.ac.uk/docs/docs.shtml