DOI: http://dx.doi.org/10.26483/ijarcs.v9i1.5500

ISSN No. 0976-5697

Volume 9, No. 1, January-February 2018

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

SEGMENTATION-FREE RECOGNITION OF URDU SCRIPT USING HMM

Prabjot Singh
Lecturer, Dept. of Computer Engineering
Govt. College of Engineering and Technology
Jammu, India

Kuljeet Singh
Lecturer, Dept. of Computer Science,
University of Jammu
Jammu, India

Jyoti Mahajan
Assistant professor, Dept. of Computer Engineering
Govt. College of Engineering and Technology
Jammu, India

Abstract: All the Urdu literature is in the form of manuscripts and typewritten books.There is a need for converting all these physical libraries into electronic libraries. Various OCRs have been developed for different languages and are widely used. Building a complete Urdu OCR is a difficult task because Urdu is highly cursive language, where ligatures overlap and style variation poses challenges to the recognition system.
We are describing a technique for automatic recognition of off-line printed Urdu text using Hidden Markov Models. Our method does not require segmentation into characters and considers each shape of Urdu character as different class resulting in a total of 196 classes (compared to 38 Urdu letters). This paper presents a novel feature extraction method based on sliding window technique, using only 16 statistical features from each sliding window thereby eliminating the need for segmentation of Urdu text. The dependency of Recognition rate of Urdu script upon, the number of states of HMM, different sizes of hierarchical window and different fonts is presented. We are using HTK (Hidden Markov Model Toolkit) for training, recognition and result analysis.

Keywords:Naskh, OCR, HMM, HTK

1. INTRODUCTION

Optical character recognition, abbreviated as OCR, is the technique that converts scanned images of handwritten, typewritten or printed text into the machine-encoded form that can be processed, edited, searched, saved, and copied for an unlimited number of times without any degradation or loss of information using a computer.
The cursive nature of Urdu language is the main challenge in its automatic recognition. Segmenting the script into characters is very difficult and complex procedure. Moreover, it always generates errors, resulting in low recognition rates. We are presenting a new method for off-line recognition of Urdu script. The method does not require segmentation into characters and is applied to cursive Urdu script, where ligatures, overlaps and style variation pose challenges to the recognition system. The method trains a single hidden Markov model (HMM) with the structural features extracted from the manuscript images. The HMM is composed of multiple character models where each model represents one letter of the alphabet. The text image is processed line by line without segmenting the line into words or characters. The horizontal position at the line image is used as an independent variable to generate simple statistical features. The performance of the proposed method on different fonts of Urdu Naskh script is assessed using samples extracted from a historical typewritten manuscript.
Character recognition for Urdu script faces challenges mainly due to its characteristics like cursive nature, multiple fonts, context-dependent shapes of characters and their position with respect to the baseline. These obstacles have played an important role in delaying character recognition system for the Urdu language compared to other languages like Latin and Chinese. Segmentation of Urdu text is proven to be a difficult and error-prone task so many researchers have proposed a Global approach for character Recognition without segmenting the words or ligatures into characters.

In 2007, M.S Khorsheed [1] proposed a method for recognizing the handwritten Arabic text images. He divided the document image into text images. It then extracts a set of simple statistical features that are injected into Hidden Markov Toolkit (HTK) for training. The System depends on the technique of character models and grammar from training samples and achieved the recognition rate of 85% for Arabic samples. It is lexicon free and uses basic units during recognition.This offer open vocabulary recognition, however, this method is expensive in terms of computation and time during feature extraction and training.
The research reveals that segmentation of words into characters is difficult in languages such as Arabic and Urdu [2]. Moreover, if the segmentation is performed it is very difficult to recognize these segmented parts, so the segmentation free techniqueshave been used by various researchers, yielding encouraging results.In 2006, G.S Lehal and Chandan Singh presented a multi-font Gurumukhi OCR for printed text with an accuracy of more than 97% [3].

2. PROPOSED SYSTEM

This section presents our proposed Recognition system in detail. Figure: 1 shows a block diagram of this system. Once the text page is scanned, the preprocessing stage reduces the

noise from the text image. The lines of text are located from the text image and simple features are extracted from each line image. The preprocessing and feature extraction stages are identical for training and recognition procedures. In training procedure, the system takes as input, the sequence of symbols coupled with the corresponding ground truth and estimates the parameters of character models. In the recognition procedure, the system takes the sequence of symbols as input to find the character sequence that has the highest likelihoods.

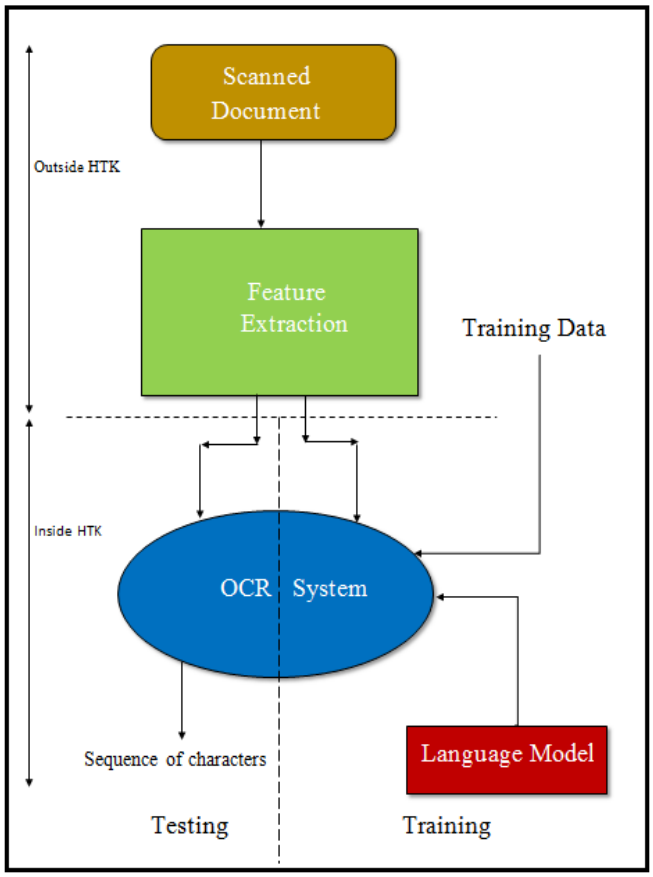


Figure1: OCR System Design

1. Data Corpus:

To train and test the OCR System, we built a data corpus consisting 4250 lines of Urdu text. Building data corpus is an important step to assess the performance of the recognition system.

2. Data Labeling:

This is also known as Transcription. Once the softcopy of data corpus is built, each Urdu character is then represented by two hexadecimal numbers. Urdu characters are context sensitive. A character can have up to four different shapes depending upon their position within a word. The total number of labels in the system equals to 196 and it includes Urdu characters, numbers, space character and punctuation marks. Labeling procedure is done manually.

3. Pre-processing:

Once the data corpus is built, the next step is to convert this data corpus into text images using a scanner or an image processing tool. The main function of this stage is to enhance the text images by reducing noise before any further processing.

4. Noise Reduction:

Noise is an important challenge to any OCR system. It may come from bad scanners or from poor documents. To minimize noise a scanner of high quality should be used. When the noise is detected in text image filter can be used to minimize the noise, but if there is still a noise in the image the image is rescanned again or discarded from data corpus.

5. Feature Vector Extraction Method:

In feature extraction phase, each segmented line of the text image is divided into a number of frames. In our approach, we take the frame width and height to be 3-7 and 64-80 respectively. The frame width and height are chosen according to our statistical analysis.

In order to use the stochastic procedure such as HMM, the feature vector should be extracted as a function of an independent variable. In speech recognition, the feature vectors are extracted from speech signals using time as an independent variable [4]. However, in offline text recognition system, the whole page needs to be recognized at a time; hence time cannot be treated as an independent variable in this case. Now assuming the horizontal axis along the text line is the independent variable, sliding window is scanned over the line from right to left as shown in Figure 2.

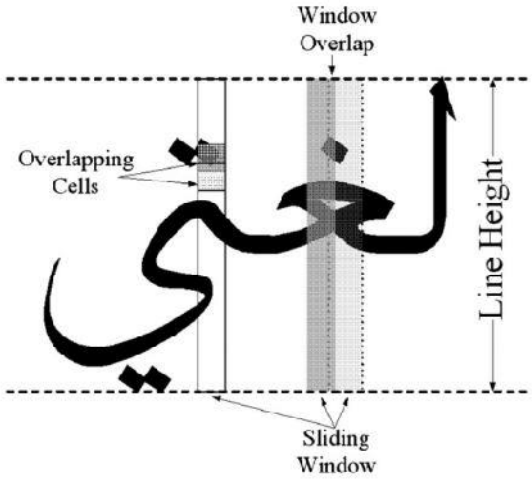


Figure2: Sliding window and overlapping cells [4]

As already mentioned, there are several methods of feature extraction such as statistical (DCT, DWT, Sliding window), structural andglobal methods [5]. In these methods, statistical features are extracted from each segmented image. These types of features are easy to compute and take a short processing time as compared to structural features. However, using this technique we are extracting only 16 simple features (of one type) per vertical strip (window) as shown in fig. 3. As the line image is already converted into the binary image (0&1). A virtual window of variable width and constant height (height of the image) slides over the entire length of the image from left to right. Each window consists of eight vertical cells dividing it into eight parts. Features (F1-F8) are then converted into sixteen features (F1-F16) as shown in fig. 3.

F9=F1+F2,
F10=F3+F4,
F11=F5+F6,
F12=F7+F8,

F13=F9+F10,
F14=F11+F12,
F15=F10+F11,
F16=F13+F14.

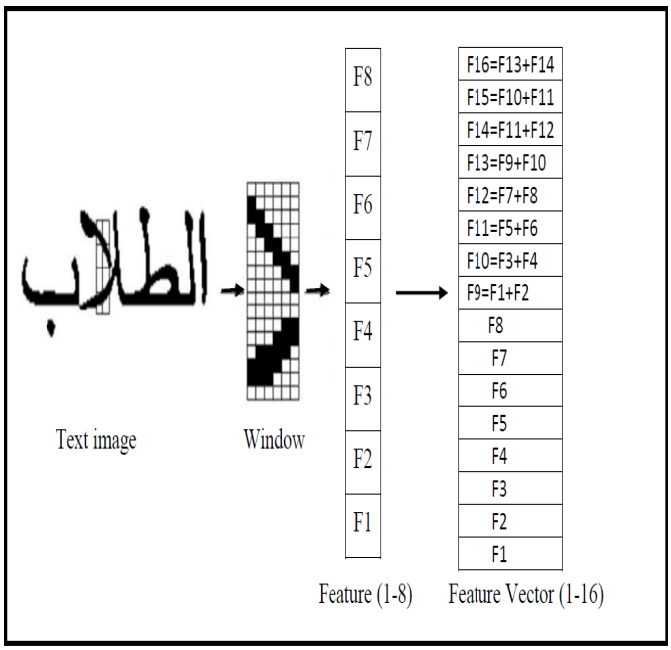


Figure 3: Feature extraction method

In many experiments, we tried different values for the window size (width and height) and overlapping (vertical and horizontal). These feature vectors of the entire image are injected into the HMM during the training process. Once the HMM is trained, the feature vectors of the image to be recognized are used as input to HTK.

6. HMMStructure:

We are representing each Urdu character, number or punctuation mark is by a distinct left-to-right HMM, as shown in fig. 4. The total number of HMMs is 196. Different shapes of an Urdu character are represented by the same character model.
To build an HMM, following parameters are required:

a) *The Number of States:* There is no mathematical way to define the best number of states to represent a Hidden Markov Model (HMM). The only way is to try the different number of states per each HMM. For simplicity, we used the same number of states per HMM for all character models.

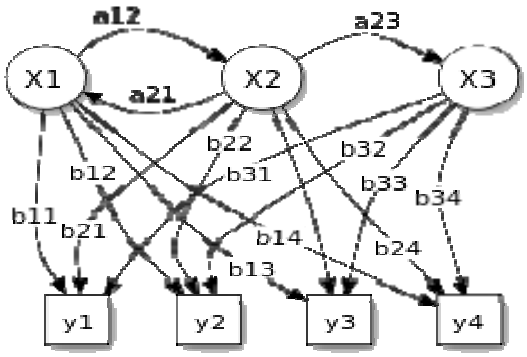


Figure 4: Hidden Markov model

Each label in the data corpus is represented by a single left-to-right HMM model with the same number of states per model [6].

b) Discrete probabilities for each emitting state: Each state 'j' in a discrete Hidden Markov Modelhas an associated observation probability distribution 'bj(ox)' that determines the probability of generating an observation ox at the position x in the line image[7]:

$$bj\ (ox) = Pj\ [v(ox)]$$

Where v(ox) is the output of the vector quantizer, given the input vector 'ox' and Pj[v] is the probability of state j generating symbol v. The output distribution consists of a table giving the probability of each possible observation symbol at each state. Each symbol is identified by a codebook index in the range 1 to M.

c) Transition Matrix: Each pair of states i and j has an associated transition probability as a_{ij}. There is a link between two states i and j if a_{ij} represented by a non-zero value in the transition matrix.

7. Training Procedure:

Defining the structure of Hidden Markov Model(HMM) is the first step towards building the recognizer. The second step is to estimate the parameters of the Hidden Markov Model(HMM) from examples of the data sequences. This process is called training procedure.

Training procedure initializes the transition probabilities and the discrete output distributions. This is done in three steps: first, the training examples are uniformly segmented. Each segment counts the number of occurrences of each symbol. These counts are normalized to provide the output distribution of each state. The Viterbi algorithm [8] discussed earlier, is applied to re-segment the data and the parameters are recomputed. The basic principle of this step depends on the concept that an HMM generates the observations. Every training example is viewed as the output of the HMM, whose parameters has to be estimated.
The second step of the training procedure is to use Baum-Welch re-estimation algorithm [9] discussed earlier, in place of Viterbi training to find the probability of being in each state at each time t using the Forward-Backward algorithm discussed earlier. The Viterbi algorithm makes a hard decision; the training vector was generated by which state.

Baum-Welch re-estimation algorithm takes a soft decision; the probabilities of being in each state at each time as discussed. The previous two steps are applied to Hidden Markov Model(HMM). This HMM, will be used in the last step as a seed for all character HMMs. In the last step, each line image is trained with the associated label file which gives the transcriptions in this line. The associated transcription constructs a composite HMM, which covers all the models in the line. The Forward-Backward algorithm is then applied and new parameter estimates are computed and a new updated HMMis outputted.

7. ***Recognition Procedure:***

Once the OCR system is built and tuned with training examples, it can be used to decode unknown feature vectors to the corresponding Urdu characters. The recognition network includes a set of nodes which are connected by arcs. Each node is represented by an HMM which is itself a network of states connected by arcs. The output of the recognition system is a sequence of Urdu characters. The system is lexicon-free where the set of characters between two consecutive spaces is represented by Urdu word without searching in any lexicon. This level is chosen because of following reasons:

1) The system can recognize any Urdu word while if present in the dictionary, some Urdu words not belonging to the dictionary may occur. This problem is named in recognition: Out of Vocabulary problem (OOV) [10]. At the HMM models level, this problem will not appear because any Urdu word consists of a limited set of Urdu characters. These characters are represented by a set of HMMs.

2) The processing unit in our OCR system is the line image. Each line image consists of a limited set of Urdu characters which is represented by a set of HMMs.

To recognize an input line image, the line image is transferred into a sequence of observations. If the number of observations in the line image is T, then every path from the start node to the end node in the recognition network, which passes through exactly T HMM states, may represent the target line. These paths have a probability which is computed by summing the probabilities of each state generating the corresponding observation and the probability of the transition from one HMM model to another. The probability of states is determined from the HMM parameters and the probabilities between HMMs are determined by the language model likelihoods.

The recognizer has to find those paths through the network which have the highest probability. These paths are found using a Token Pass Algorithm. This algorithm works as follows:

1. At horizontal position 0 in the line image, a token is placed at the start of every HMM model in the network.

2. At each horizontal position step, propagate the token along connecting transitions and stop at an emitting HMM state. If there is more than one exit from the node, copy a new token for every path

3. Increment the probability of each token by $a_{ij} + b_j$ (Ox), where a_{ij} is the transition probability from the state 'i' to state j, and b_j (Ox) is the discrete probability of observation Ox in state j.

4. At the end of each transition step, all tokens are discarded except the token with the highest probability

5. Each token that passes through the recognition network must save a history recording its route.

When the token comes to the last transition, the route of the token with the highest probability is the output of the recognizer.

This comparison is performed using dynamic programming to align the two transcriptions and then count the number of:

1. Substitution Errors (S): The number of wrong labels that substitutes correct labels in the recognizer output sequence

2. Deletion Errors (D): The number of correct labels deleted from the recognizer output sequence.

3. Insertion Errors (I): The number of wrong labels inserted between two consecutive correct labels in the recognizer output sequence.

4. Total labels (N): The total number of labels in recognizer output sequence.

Once the optimal alignment has been found the Correction rate (% Correction) is then:

$$\% \text{ Corr} = \frac{N-D-S}{N} \times 100$$

If H is the number of correct labels, then H = N – S – D. Then the correction rate becomes:

$$\%\text{Corr} = H/N \times 100$$

The accuracy (Acc.) is then defined as:

$$\text{Acc} = (H-I)/N = (N - S - (D - I))/N$$

3. EXPERIMENTAL RESULTS

In this section, the performance of the Urdu character recognition system presented is evaluated. The effect of various parameters like the number of states of HMM, number of horizontal and vertical overlaps during feature extraction phase on the recognition rate of Urdu text has been discussed. Moreover, the performance of the system on Naskh (Naskh-Andalus, Naskh-Arial, Tahoma fonts) font in the same data corpus, codebook size and on different training and testing sets are also assessed.

1) *The Number of States/HMMs:*Once the best combination of frame width and number of cells per frame is selected using sliding window, the next important parameter to determine if the number of states/HMM. There is no mathematical way to find the best number of states/HMM. In this experiment, the different number of states/HMM are tested, namely: 4, 5, 6, 7, 8 and 10 states/HMM. For simplicity, all HMMs have these same number of states. From Table 1, 6–states/HMM is the best number of states/HMM. It gives the best correction rate = 93.39%.Arial font is used in this experiment for determining the best combination of States/HMM. It is being observed that system performance falls drastically when 8 states or more are used.

2) *Cells of different Size and type:*To study this approach, the first thing we need to find is the best Window size that will give a good performance and consumes an acceptable processing time.

Table 1: Different number of states/HMM

Number of states	H	D	S	I	%Acc	%Cor
5	66175	403	6561	7525	80.19	90.48
6	**68304**	**634**	**4201**	**3401**	**88.74**	**93.39**
7	67390	1631	4118	704	91.18	92.14
8	41089	8468	23582	469	55.54	56.18

Here, there are twotypes of cells: with and without overlap. The overlap between cells is in the vertical direction while the overlap between frames is in the horizontal direction. The overlap operation increases the number of frames (i.e., data) generated from the line image. The disadvantage is that ittakes more time to generate the feature vectors. The different combination of cell size and type that are used in this approach are as follows:

a) Window without any type of overlap (H0V0).

b) Window with one horizontal pixel overlap and one vertical pixel overlap (H1V1).

c) Window with one horizontal pixel overlap and three vertical pixels overlaps (H1V3).

d) Window with two horizontal pixels overlaps and one vertical pixel overlap (H2V1).

e) Window with two horizontal and verticalpixels overlaps (H2V2.).

From the experiment, we note that when the frame width increases the system performance decreases. This is because the use of large frame width leads to a low number of frames and hence low data to HMM model. Further, we also concluded that the system gives optimum performance when frame width is 3with 1 horizontal and 3 vertical overlaps.The Table 2. Below show the performance of each type of overlap when the number of states of HMM is 6.

Table 2: Performance with different overlapping

Frame width	Overlap (horizontal vertical)	H	D	S	I	N	%Acc.	%Correction
3	H1V0	9755	103	996	749	10854	82.97	89.87
3	H1V2	9760	96	96	726	10854	83.5	90.20
3	**H1V3**	**13419**	**126**	**1092**	**958**	**14637**	**85.13**	**91.68**
4	H2V2	9825	94	935	708	10854	84.00	90.52
3	H2V3	10769	152	3716	3770	14637	47.82	73.57

3) *Performance on different Training:* In this section, we studied the effect of the number of training set on the performance of the recognition system. Here we try different training sets consisting of 1500, 2500, 3200 and 4000 text line images. The 'Andalus' font with the window size of 3 pixels, 1 horizontal and 3 vertical overlaps (w3_H1V3) are used.Table 3 showed that performance of system increases when the training sets are increased. The highest performance occurs when the system was trained with 3200 line images. It gives a correction rate equal to 92.11% and Accuracy of 85.93%.

Table3: System Performance when different numbers of training samples were used.

Training set	H	D	S	I	%Acc	%Cor
1500	3917	1645	5292	820	28.53	36.09
2500	13419	126	1092	958	85.13	91.68
3200	**13483**	**128**	**1126**	**905**	**85.93**	**92.11**
4000	13250	126	1261	898	84.00	90.52
4250	9760	96	96	726	83.5	90.20

4) *Results on different Fonts:* Five different Urdu fonts were used for recognition and testing (viz. Arial, Tahoma, Akhbar, Naskh, and Andalus).Table4 summarizes the results of Akhbar, Andalus, Naskh and Arial fonts.

Table 4: Summary of results on different fonts (N= 73139)

Font	H	D	S	I	%Acc	%Cr
Akhbar	68300	368	4471	1798	90.93	93.38
Andalus	65238	533	7368	5278	81.98	89.20
Naskh	66704	485	7121	4233	85.41	90.79
Arial	**68305**	**638**	**4196**	**3404**	**88.74**	**93.39**

4. ERROR ANALYSIS

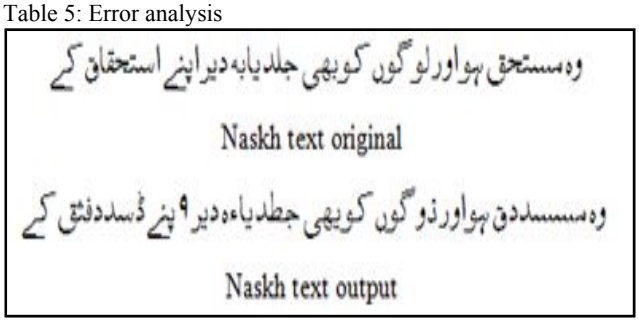
Although experimental show up to 93.39% accuracy in the best case there are some situations when our system does not perform as expected. Figure.4 shows one such case, as we can see some of the characters are misinterpreted by our

system.Wehave seen there are differences in case of the word and more common characters. The placement of single dot canchange the entire meaning. Even dots are increased by the

Figure 4: Sample output.

presence of noisy input, thereby affecting the accuracy of the system. Table 5 below presents few examples of such words or characters.

Table 5: Error analysis



Original word	Recognized Word	Error details
ت	ث	Extra dot in output
جلد	جطد	Loop formed in output
گ	گخ	Extra character added to original word
شش	دسن	Dots removed and shape changed

5. FUTURE WORK AND CONCLUSION

1. *Future Work:*

There are many extensions that can be done either to enhance the performance of the system or to make the approach applicable to a wider range of tasks related to Urdu text Recognition. These extensions are as under:

a) *Switch to handwritten Urdu script:*The implemented system recognizes typewritten Urdu script. If a data corpus of handwritten Urdu script is built, the training and recognition procedures implemented here will be applied to recognize handwritten script. The only difference will be at the preprocessing stage.

b) *Using different types of HMMs***:**Since the implemented recognition system uses discrete HMMs because it is a fast system, continuous HMMs or tied-mixture HMMs can be used instead. The use of HTK makes this job very easy because HTK supports all those types.

c) *Using new novel features***: T**he statistical features that were used here gave good results, but new novel statistical features can be used instead to reduce the confusion between different characters.

d) *Using degraded and noisy data:*Instead of using printed pages, we plan to test the system using noisy data such as fax and nth-generation photocopies.

e) *Multi-language system: S*ince the implemented system is segmentation-free, this system can be extended to recognize other language words inside the Urdu text.

2. *Conclusion:*

In our work, we addressed the problem of automatic reading of Urdu typewritten script. Urdu script presents important challenges such as cursiveness and the Urdu characters are context sensitive to their location within the word. The segmentation of Urdu words into characters is a hard job and always is a point of failure in the segmentation-based systems. We proposed a segmentation-free system to improve the recognition performance especially when the Urdu words are not easily separable.

The central model of the proposed system is the hidden Markov models. Each character, with its different shapes and from, was represented by a distinct HMM. The system was built on Hidden Markov Models Toolkit (HTK) which had been designed primarily for speech recognition research.

The system presented in this paper is the Urdu OCR system built on the HTK. The preprocessing and feature extraction were performed outside the HTK while the training and recognition procedures were performed inside HTK. The implemented system is a lexicon-free system. It can handle unlimited Urdu vocabularies since the system works on the character level. The output of the recognizer is a sequence of characters (Unicode).

In order to use HMMs, the horizontal position at the line image was assumed to be the independent variable to extract the statistical features. Each line image was scanned from right-to-left with a narrow vertical window which was divided horizontally into eight cells and at each horizontal position; simple statistical features were computed from pixels falling within that window. These features are then injected into the HTK for training.

The System is trained with 4250 lines from the data corpus and tested with 150 images randomly chosen from data corpus. The system showed the correctness of 92.11% and Accuracy of 85.93% when experimented with Andalus Font. Experimental results showed the dependence of system performance on the number of states per HMM. We also observed that the performance increases when it is trained with more data. But increasing the test cases does not affect the system performance.

REFERENCES

[1] Khorsheed, Mohammad S. "Offline recognition of omnifont Arabic text using the HMM ToolKit (HTK)." Pattern Recognition Letters 28.12 (2007): 1563-1571.

[2] AL-MUHTASEB, H.A., MAHMOUD, S.A. and QAHWAJI, R.S., 2008. “Recognition of off-line printed Arabic text using Hidden Markov Models.” Signal Processing, Elsevier (12), pp. 2902-2912, 2008.

[3] Lehal, Gurpreet S. "A Complete Machine-Printed Gurmukhi OCR System." In Guide to OCR for Indic Scripts, pp. 43-71. Springer, London, 2009.

[4] Khorsheed, Mohammad S."Recognising handwritten Arabic manuscripts using a single hidden Markov model." Pattern Recognition Letters 24.14 , 2235-2242. (2003).

[5] Nefian, A. V., & Hayes, M. H. (1998, May). Hidden Markov models for face recognition. In Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on (Vol. 5, pp. 2721-2724). IEEE.

[6] Bazzi, Issam, Richard Schwartz, and John Makhoul. "An omnifont open-vocabulary OCR system for English and Arabic." Pattern Analysis and Machine Intelligence, IEEE Transactions on 21.6 (1999): 495-504.

[7] Young, S., Evermann, G., Gales, M., Hain, T., Kershaw, D., Liu, X., Moore, G., Odell, J., Ollason, D., Povey, D. and Valtchev, V., 2002. The HTK book. Cambridge university engineering department, 3, p.175.

[8] Forney GD. The viterbi algorithm. Proceedings of the IEEE. 1973 Mar;61(3):268-78.

[9] Baggenstoss, Paul M. "A modified Baum-Welch algorithm for hidden Markov models with multiple observation spaces." IEEE Transactions on speech and audio processing 9, no. 4 (2001): 411-416.

[10] Lehal G. A word segmentation system for handling space omission problem in urdu script. InProceedings of the 1st Workshop on South and Southeast Asian Natural Language Processing 2010 (pp. 43-50).