An Arabic Optical Character Recognition System Using Restricted Boltzmann Machines

Abdullah M. Rashwan, Mohamed S. Kamel, and Fakhri Karray

University of Waterloo

Abstract. Most of the state-of-the-art Arabic Optical Character Recognition systems use Hidden Markov Models to model Arabic characters. Much of the attention is paid to provide the HMM system with new features, pre-processing, or post-processing modules to improve the performances. In this paper, we present an Arabic OCR system using Restricted Boltzmann Machines (RBMs) to model Arabic characters. The recently announced ALTEC dataset for typewritten OCR system is used to train and test the system. The results show a 26% increase in the average word accuracy rate and 8% increase in the average character accuracy rate compared to the HMM system.

1 Introduction

Digitizing information sources and making them available for the Internet users is taking much attention these days in both academic and industrial fields. Unlike typewritten Latin OCRs, typewritten OCRs for cursive scripted languages (ex. Arabic, Persian, etc.) still encounter a plethora of unsolved problems. The best average word accuracy rate achieved for large vocabulary Arabic OCR according to the rigorous tests done by ALTEC organization on 3 different commercial OCRs in 2011 was slightly below 75%. Providing a decent solution for recognizing cursive scripted language will allow millions of books to be available on the Internet, it will also push Arabic Document Management Systems (DMSs) steps forward.

There are few papers that tackled the typewritten Arabic OCR problems. The lack of dataset is one of the main reasons that directed the researchers away from tackling this problem. In 1999, [1] designed a complete Arabic OCR system, they used a good dataset to train and test their system. The dataset however is biased towards magazine documents. They reported a character error rate of 3.3%, such high accuracy is not only a result of using HMM classifier, but also because they supported the classifier with strong preprocessing and postprocessing modules to boost the accuracy. In 2007, [2] designed an Arabic multi-font OCR system using discrete HMMs along with intensity based features. Character models were implemented using mono and tri models. He achieved a character accuracy rate for the Simplified Arabic font of 77.8% using mono-models. In 2009, [3] introduced new features to use along with discrete HMMs to model Arabic characters. These features are less sensitive to font size and

style. They reported character accuracy rate of 99.3% which is very high, such high accuracy is due to the use of a synthesized and very clean dataset, and using high quality images scanned at 600dpi. In 2012, [4] developed the previous system using a practical dataset created by ALTEC. The reported character accuracy rates for the HMM system using bi-gram and 4-gram character language model were almost 84% and 88% respectively. The paper introduced also an OCR system using Pseudo 2D HMM achieving a CER of 94% and gaining more than 6% over using HMM. Most of the work done so far to tackle the problem of recognizing Arabic letters, and cursive scripted languages in general, lacks either a good dataset to build a practical model or a good model to come up with a decent solution. In this work, a good model along with a practical dataset have been used to overcome the limitations and drawbacks of the previous systems.

In the following sections, Sec. 2 presents the system architecture and the feature extraction, Sec. 3 presents HMMs and how to apply them on the Arabic OCR problem, Sec. 4 presents the RBMs and how they can improve the performance of the HMM system, experimental results are presented in Sec. 5. The final conclusions are presented in Sec. 6.

2 System Architecture

Fig. 1. System architecture of the Arabic OCR

The architecture of our OCR system is shown in Fig. 1. In the recognition phase, first, the printed text pages are scanned and digitized. Lines and words boundaries are specified automatically using histogram-based algorithm. After automatically extracting the words, each word is segmented to vertical frames and features are extracted from each frame. A moving window of width 11 pixels and a step size of 1 pixel, is used to extract the features for each frame.

The use of 11-pixels window size is due to the fact that the average character width is ~ 11 pixel. The features are simply the row pixels, the word height is resized to 20 pixels in order to form a fixed length feature vector. Using a Viterbi decoder, the most likely characters sequence is obtained using the feature vectors sequence, the classifier model, and n-gram language model probabilities that were constructed during the training phase. The decoder output is then processed to obtain the recognized words.

In the training phase, the printed text pages are photocopied using different photocopiers. Both pages, the original and the photocopied, are scanned and digitized. Different scanners and photocopiers are used in order to represent practical noise in the training database. Digitized images follow the same steps in the recognition mode to extract the frames. Frames are then concatenated in sequence and stored in order to be used in the parameters estimation. The extracted features along with the corresponding text are used in the parameters estimation for the classifier. In this paper, three different classifiers are used in the system; HMM classifier, neural network trained using backpropagation algorithm, and neural network pretrained using RBMs. The power of the RBMs is that they can model any distribution without making assumptions that the distribution is Gaussian or discrete as in the classical HMMs. A training scheme is used to insure a good estimation of the parameters and different experiments are held to achieve the best performance. On the other hand, a corpus of ~ 0.5 Giga word is used in estimating the character-level language model. Both the language model and the classifier model are used in the recognition phase.

3 HMM and Arabic OCR

Fig. 2. The figure shows a five state HMM model of the 'أ' character and the state alignment of extracted features. Zeros in the feature vectors represent the dummy segments inserted in order to form fixed size feature vectors.

First we have to define the number of character models that we want to use. The advantage of using large number of models is that the system will be able to distinguish between different shapes easily but that requires having sufficient number of training examples per each model. One should decide the number of different models based on the training database. After deciding the number of models, we have to decide the HMM model per each character. It is common to

use first order HMM with right-to-left topology as shown in Fig. 2. The number of hidden states can be either the same for all models or each model can have different number of hidden states. Because we deal with discrete HMMs in this work, the feature vectors needs to be quantized.

The vector-quantizer serially maps the feature vectors to quantized observations. A codebook generated by the codebook maker module during the training phase is used for such mapping. The codebook size is a key factor that affects the recognition accuracy and is specified empirically to minimize the WER of the system.

Using numerous sample text images of the training set of documents evenly representing the various Arabic characters, the codebook maker creates a codebook using the LBG vector clustering algorithm. The LBG algorithm is chosen for its simplicity and relative insensitivity to the randomly chosen initial centroids compared with the basic K-means algorithm.

4 Restricted Boltzmann Machines

We use a Neural Network to replace the discrete distribution for the hidden HMM states. The Neural Network is first pre-trained as a multilayer generative model of a window of spectral feature vectors without making use of any discriminative information. Once the generative pre-training has designed the features, we perform discriminative fine-tuning using backpropagation to adjust the features slightly to make them better at predicting a probability distribution over the states of hidden Markov models.

4.1 Learning the Restricted Boltzmann Machines

As explained in [5], Restricted Boltzmann Machines (RBMs) are used to learn multi-layers of deep belief networks such that only one layer is learned at a time. An RBM is single layer and is restricted in the sense that no visible-visible or hidden-hidden connections are allowed. The learning works in a bottom-up scheme using a stack of RBM layers. The weights for the first layer are estimated using the input features, and then we fix the weights of the first layer and use its latent variables as an input to second layer. In such way, we can learn as many layers as we like.

In binary RBMs, both the visible and hidden units are binary and stochastic. The energy function of the binary RBMs is:

$$E(v, h|\theta) = - \sum_{i=1}^V \sum_{j=1}^H w_{ij} v_i h_j - \sum_{i=1}^V b_i v_i - \sum_{j=1}^H a_j h_j$$

Where $\theta = (w, a, b)$ and w_{ij} represents the symmetric interaction term between visible unit i and hidden unit j while b_i and a_j are their bias terms. V and H is the number of the visible and hidden units.

The probability that an RBM assigns to a visible vector v is:

$$p(v|\theta) = \frac{\sum_h e^{-E(v,h)}}{\sum_u \sum_h e^{-E(v,h)}} \quad (1)$$

Since there are no hidden-hidden connections and visible-visible connections, the conditional distribution $p(h|v, \theta)$ and $p(v|h, \theta)$ can be presented as:

$$p(h_j = 1|v, \theta) = \sigma(a_j + \sum_{i=1}^V w_{ij}v_i) \quad (2)$$

$$p(v_i = 1|h, \theta) = \sigma(b_i + \sum_{j=1}^H w_{ij}h_j) \quad (3)$$

Using “contrastive divergence” training procedure [6], the weights can be updated using the following update rule:

$$\Delta w_{ij} \propto \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{reconstruction} \quad (4)$$

4.2 Using RBMs in Arabic OCR

When using HMMs, the commonly used method for sequential data modeling, the observations are modeled using either discrete models or Gaussian Mixture Models (GMMs). Although these models are proven to be useful in many applications, they encounter serious drawbacks and limitations [5,7]. In this work, Neural Network (NN) will replace the discrete models or GMMs to model the inner states for the Arabic characters. The recognizer will use the posterior probabilities over the inner states from the NN combined with a Viterbi decoder in order to recognize an input Arabic word. A well-trained HMM system is used for state alignment of all training database in which feature vectors are assigned to states they belong to (Fig. 2). In the pre-training step, the Neural Network is pre-trained generatively using a stack of RBMs, this is essential for avoiding over fitting and to ensure a good convergence point in the classification step. In the classification step, a softmax layer is added to the pre-trained network then the network is trained using backpropagation algorithm. The trained Neural Network is used to predicting a probability distribution over the states of the HMM replacing the Gaussian Mixture Models (GMMs) or the Discrete Models.

5 Experimental Results and Evaluation

The system structure presented in Sec. 2 is trained using part of the ALTEC dataset [8]. Only 2 fonts, Simplified Arabic 14 and Arabic Transparent 14, are used to simplify the problem and to compare the RBMs to the HMMs. Around 4500 words scanned at 300dpi resolution are used to train both systems. We used 151 HMM models to model different Arabic character shapes. HTK toolkit

is used for models training and recognition [9]. A manually modified version of the HTK is used to recognize the character sequences using the trained Neural Network. SRILM toolkit is used to create the character based n-gram language model [10]. The test database consists of 1790 words, these pages are scanned on different scanners than the ones used for the training dataset.

We analyzed the RBMs performance by conducting four experiments. The first experiment aims at testing the effect of using a stack of RBM layers instead of directly estimating the weights using backpropagation algorithm. The second experiment tests the effect of varying the number of hidden nodes on the system performance. The third experiment aims to test the effect of varying number of hidden layers on the performance. The last experiment compares the RBM-based system to the HMM-based system in terms of accuracy and speed.

For the HMM system, we used a discrete distribution to model the hidden states. We used codebook of size 1000, and the HTK toolkit was used for the training. A forced-alignment on the state level was performed using the HMM models, then the Neural Network, pre-trained generatively using a of RBMs, was trained using the state labeling along with the corresponding feature vectors.

5.1 Effect of Pretraining Using RBMs

Table 1. Word accuracy rates for the backpropagation training algorithm and the neural network pretrained using RBMs

	Backpropagation	RBM Accuracy
Average WAR	72%	81%

Table 2. Detailed word accuracy rates when varying the number of hidden units

Number of Hidden Units	100	200	500	1000
Average WAR	61%	72%	79%	81%

Table 3. Detailed word accuracy rates when varying the number of hidden layers

	1 hidden layer	2 hidden layers
Average WAR	71.7%	72.4%

The goal of these experiments is to test the effect of using RBMs. We constructed two neural networks of 1 hidden layer and 1000 hidden nodes. We trained the first network using backpropagation directly, and we pretrained the second network using RBMs then we tuned it using backpropagation algorithm. Table 1 shows the word accuracy rates for both networks, we can clearly see a 9% increase in the accuracy which is very large. Using RBMs is even more useful when we deal with multi-layer neural networks[11].

5.2 Varying the Number of Hidden Units

In this experiment, we trained the neural network and varied the number of hidden units from 100 to 1000. The evaluation is based on the word accuracy rates. As shown in Table 2, we can see that the word accuracy rate increases as we increase the number of hidden units. The gain we obtained is not linear with increasing the number of hidden units, it takes the shape of the log curve where the accuracy saturates at certain level no matter the number of hidden units that you add.

5.3 Varying the Number of Hidden Layers

In this experiment, we compared the accuracy using 1 and 2 hidden layers. We fixed the number of hidden units to 200 and we compared a neural network with a 200 nodes layer to a neural network with a two stacked hidden layers with 100 hidden nodes each. Fig. 3 shows that we gained $\sim 0.7\%$ accuracy just by using 2-layers without changing the number of nodes. This is useful when we want to improve the accuracy without increasing the recognition time, we will however face the difficulties of tuning multi-layer network during the training phase.

5.4 Comparing the RBM and the HMM Based Systems

Table 4. Word and Character Accuracy Rates

	HMM Accuracy	RBM Accuracy
Average WAR	55%	81%
Average CAR	87%	95.2%
Recognition Time (m.sec)	26	198

In these experiments, we used a discrete HMM-based system with a codebook of size 1000. We compared the performance of this system to the RBM system where the number of hidden nodes is also 1000. For both systems we used the same features and the same bigram language model, and the codebook size for the HMM system equals to the number of hidden nodes for the Neural Network system. The system performance is evaluated using word accuracy rate. As shown in Table 4, the performance of the Neural Network system is much higher than the HMM system. This is due to the fact that the HMM is a generative model that tries to maximize the likelihood of the data. On the other hand, the Neural Network is pretrained using RBMs, then a fine tuning discriminative training is performed using the backpropagation algorithm. Although the word accuracy rate is more practical and intuitive, the Character Accuracy Rate (CAR) is also important and the results are shown in Table 4.

Of course there is something we should pay for such improvement, the computational complexity of the Neural Network based system is higher than the one for the discrete HMM. The discrete HMM simply stores the probabilities in

lookup tables, that makes it too fast to evaluate the hidden state distributions. The Neural Network system has to evaluate the output of the 1000 hidden nodes plus the output of the 1384 output nodes. The recognition time per character for both systems is listed in Table. 4.

6 Conclusion and Future Work

Recently, research has been directed to improve the inputs and outputs (ex. preprocessing, features, and postprocessing) to the HMM model other than to improve the character modeling. In this paper, we have made use of RBMs to model Arabic characters. The experimental results looks very promising compared to a baseline HMM system. For future work, using more set of features can be used instead of the row pixels, also more font sizes and styles will be involved in the training.

References

1. Bazzi, I., Schwartz, R., Makhoul, J.: An omnifont open-vocabulary ocr system for english and arabic. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21, 495–504 (1999)
2. Khorsheed, M.: Offline recognition of omnifont arabic text using the hmm toolkit (htk). *Pattern Recognition Letters* 28, 1563–1571 (2007)
3. Attia, M., Rashwan, M., El-Mahallawy, M.: Autonomously normalized horizontal differentials as features for hmm-based omni font-written ocr systems for cursively scripted languages. In: 2009 IEEE International Conference on Signal and Image Processing Applications (ICSIPA), pp. 185–190 (2009)
4. Rashwan, A.M., Rashwan, M.A., Abdel-Hameed, A., Abdou, S., Khalil, A.H.: A robust omnifont open-vocabulary arabic ocr system using pseudo-2d-hmm. In: *Proc. SPIE*, vol. 8297 (2012)
5. Mohamed, A., Dahl, G., Hinton, G.: Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech, and Language Processing* 20, 14–22 (2012)
6. Hinton, G.: Training products of experts by minimizing contrastive divergence. *Neural Computation* 14, 2002 (2000)
7. Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., Kingsbury, B.: Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine* 29 (2012)
8. Altec database (2011), http://www.altec-center.org/conference/?page_id=84
9. Young, S.J., Evermann, G., Gales, M.J.F., Hain, T., Kershaw, D., Moore, G., Odell, J., Ollason, D., Povey, D., Valtchev, V., Woodland, P.C.: *The HTK Book*, version 3.4. Cambridge University Engineering Department, Cambridge, UK (2006)
10. Stolcke, A.: SRILM – an extensible language modeling toolkit. In: *Proceedings of ICSLP*, Denver, USA, vol. 2, pp. 901–904 (2002)
11. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. *Science* 313, 504–507 (2006)