MULTI LEVEL HMM FOR HANDWRITTEN WORD RECOGNITION

Mou-Yen Chen†and Amlan Kundu‡
†ITRI, Hsinchu, Taiwan, ROC.
‡US WEST Advanced Technologies, Boulder, CO 80303.

ABSTRACT

In this paper, we have introduced a novel approach for handwritten word recognition using Multi-Level Hidden Markov Models (MLHMM). The MLHMM is a doubly embedded network of HMM's where each character is modeled by an HMM while a word is modeled by a higher-level HMM. In the character model, we associate the observation with the transition. By introducing the technique called 'tied transition', i.e., the segments which have the same semantic meaning will be 'tied' together, we have successfully built up the character model by an HMM with 4 states, 5 observations (or symbols) and 7 transitions (Fig. 2(a)). Thus, as the states are not assigned any semantic meaning, the re-estimation algorithm is applicable. At the character level, the best model is chosen as the recognition result. So, the character model is purely a model discriminant HMM (MD-HMM) based approach. For the word model, on the other hand, both the MD-HMM and the path discriminant HMM (PD-HMM) [1] approaches are used and their respective performances are demonstrated.

1 Introduction

In our previous approaches, the models are actually semi-hidden Markov models [1, 2], i.e., the states of HMM's are transparent during training. Because the re-estimation algorithm such as the Baum-Welch algorithm [3] does not preserve the correspondence of the states to their semantic meanings, it is not suitable for the semi-hidden Markov models. The model parameters, therefore, are estimated by counting the occurrences of states, symbols, etc., from the labeled training samples. In this paper, we have introduced a novel approach for handwritten word recognition using Multi-Level Hidden Markov Models (MLHMM). The MLHMM is a doubly embedded network of HMM's where each character is modeled by an HMM while a word is modeled by a higher-level HMM. The major difference between the new system and the previous approaches is the *output-independence assumption* of the HMM [4]. In this new model, we associate the observation with the transition. As the states are not assigned any semantic meaning, the re-estimation algorithm is applicable.

*The authors could be reached using the internet address akundu@advtech.uswest.com .

2 HMM Topology and Training

Using the segmentation algorithm [2, 4], most of the characters are segmented into three segments or less. For cursive writing, the characters are usually linked by some meaningless strokes – ligature. Based on these observations, we have found the character model as shown in Fig. 2(a) to be quite suitable. The five observation symbols in character models are:

W : the whole characters; L, M, R : the left, the middle, and the right parts of the characters; N : null segments, i.e., ligatures.

For the word model, in MD-HMM approach, we cascade the character models while retaining only one N observation between characters, as shown in Fig. 2(b).

The following training procedures are applied to 26 character models individually.

2.1 Initialization

By examining the segmentation results of each character, we manually characterize the segments as W, L, M, R, or N. The transition probability a_{ij} is initialized by calculating the occurrences of pairs of states. That is,

$$a_{ij} = \frac{\text{no. of transitions from } s_i \text{ to } s_j}{\text{no. of transitions from } s_i} \quad 0 \leq i, j \leq 3 \tag{1}$$

All the models start with the initial state s_o. To estimate the symbol probability $b_{ij}(k)$, we first apply the k-means clustering algorithm (with fixed SNR) to group the training samples (feature vectors of segment images [1,4]) into several clusters. Please note here that we have used a global codebook for all the character models. The symbol probability for each character model can be initialized by counting the oc-

0-7803-2431-5/95 $4.00 © 1995 IEEE

currences of the symbols in each cluster. More precisely,

$$b_{ij}(k) = \frac{\text{no. of times } symbol(i,j) \text{ appears in cluster } k}{\text{no. of times } symbol(i,j) \text{ appears}} \quad (2)$$

where $symbol(i,j)$ maps the observation emitted at transition $i \rightarrow j$ to one of $\{ W, L, M, R, N \}$. We will refer to $b_{i,j}(k)$ as the probability of observing symbol k with state transition $i \rightarrow j$. With these initially estimated parameters, the Baum-Welch algorithm is then applied to increase the likelihood of the models using their representative training sequences.

2.2 Model Normalization

In the recognition phase of MD-HMM, we can simply classify the given observation sequence to the class whose model has the largest best-path likelihood. That is,

$$w^* = \arg \max_{1 \leq w \leq W} P_w \theta_w \; ; \quad P_w = P(I_w^*, O \mid \lambda_w) \quad (3)$$

P_w is the best-path likelihood of model λ_w given the observation sequence O, and this likelihood can be found by the Viterbi algorithm [3]. The problem, basically, is to find a weight factor θ_w associated with each model λ_w such that the overall recognition result (based on training set) is optimal. We choose the range for the weight factors to be [-M,M], and sequentially optimize for each HMM starting from the model that has the worst recognition performance. Although this algorithm does not guarantee the optimal set for the weight factors, it does ensure that the overall recognition rate of the training set would increase. More details can be found in [4].

2.3 Co-occurrence Smoothing

Compared with the number of free parameters, the number of training data is usually insufficient. This may result in inaccurate estimation of the HMM statistics. In practice, smoothing of the parameters after re-estimation is essential if enough training data are not available [5]. The co-occurrence method is one such smoothing technique to achieve better performance.

Co-occurrence smoothing is based on the formula of computing $P_{co}(u \mid v)$, the co-occurrence probability of symbol u appearing given symbol v. By definition,

$$P_{co}(u \mid v) = \frac{\sum_{w=1}^{W} \sum_{i=1}^{N} \sum_{j=1}^{N} a_{ij}^w b_{ij}^w(u) b_{ij}^w(v) P(w)}{\sum_{k=1}^{M} \sum_{w=1}^{W} \sum_{i=1}^{N} \sum_{j=1}^{N} a_{ij}^w b_{ij}^w(k) b_{ij}^w(v) P(w)} \quad (4)$$

Here, W is the total number of HMM's; N is the number of states for each HMM; M is the total number of clusters. $a_{ij}^w, b_{ij}^w(k)$, and $P(w)$ are the transition probability, symbol probability, and the *a priori* probability for model λ_w. $P_{co}(u \mid v)$ can be loosely interpreted as "*when symbol v is observed, how often symbol u is observed in similar contexts*". These probabilities are obtained from the parameters after re-estimation, except $P(w)$ which is usually assumed to be uniformly distributed. By using the co-occurrence probability $P_{co}(u \mid v)$, smoothed parameters can be obtained as

$$\bar{b}_{ij}^w(u) = \sum_{k=1}^{M} P_{co}(u \mid k) b_{ij}^w(k) \quad (5)$$

$$b_{ij}^w(k) = \lambda b_{ij}^w(k) + (1-\lambda)\bar{b}_{ij}^w(k), \quad 0 < \lambda < 1 \quad (6)$$

The best λ is determined empirically (0.8).

3 Recognition

Given the word image, a sequence of segment images is obtained from the result of the sub-character segmentation algorithm proposed in [2, 4]. The combination feature set described in [1] is first used to represent each segment image as a 35-dimensional feature vector. Each feature vector is then assigned its symbol number by the nearest-neighbor classifier with the codebook generated during the training phase. Finally, this sequence of symbols and the associated dictionary for this word are applied to the word model. Both the HMM strategies (MD-HMM and PD-HMM) are used in the word models. Fig. 1 shows the system diagram.

3.1 MD-HMM Approach

Here, we need to build up an HMM for every word in the dictionary. The word model is constructed by cascading the character models, as shown in Fig. 2. While the state transition (A) and symbol (B) probabilities are obtained from each of the individual character models, the initial state (Π) and the last state (Γ) probabilities for word model can be simply assigned as

$$\pi_i = \begin{cases} 1.0 & \text{if } s_i = F_i \\ 0.0 & \text{else} \end{cases} \quad (7)$$

$$\gamma_i = \begin{cases} 1.0 & \text{if } s_i = L_i \\ 0.0 & \text{else} \end{cases} \quad (8)$$

Here, F_i is the first state of the first character in the word; and L_i is the last state of the last character in the word. With the given sequence of observation symbols, the best-path probability for each word model can be computed using the Viterbi algorithm. This probability is then normalized (all in *log* form)

$$\mathcal{L}(w) = P_w + \theta(w) \quad (9)$$

where $\theta(w)$ is the weight factor for word w; and this probability is obtained by averaging the weight factors from every character model in this word. The word model which leads to the highest normalized likelihood will be chosen.

3.2 PD-HMM Approach

The word recognition algorithm for our PD-HMM approach is based on the level-building algorithm [3].

As there is no dictionary involved in the level-building algorithm, the output character strings may not be meaningful, i.e., they may not be included in the dictionary. Thus, the string match procedure as described in [1,4] is used here as a postprocessing procedure to rank the dictionary entries. Also, we extend the level-building algorithm to give K candidate words instead of only one choice using the strategy similar to the parallel version of MVA described in [1]. Our level-building algorithm is executed as follows. Given an observation sequence $O = o_1, \ldots, o_T$, for all $\ell \leq T$, we need to find the probabilities for the ℓ-character strings. each of which corresponds to a ℓ-character path in the dictionary tree. The decision about the output strings is made by comparing these probabilities. However, an exhaustive search of all possible paths will result in huge computations for a long observation sequence. Instead, our search is done layer by layer. At each layer, only K nodes, which have the K most probable paths reaching this layer, are retained. The number K can be fixed or dynamically determined according to the number of nodes in this layer. More explicitly, let us define $B_t^\ell(k)$ as the k-th best probability at layer ℓ given observation $o_1, \ldots, o_t$. Thus, $B_t^\ell(k)$ can be found recursively as

$$B_t^\ell(k) = k - th \max_{\substack{\ell-1 \leq l \leq t-1 \\ 1 \leq j \leq K \\ n \in A_l^{\ell-1}(j)}} B_l^{\ell-1}(j) P(n \mid O_{l+1}^t) \qquad (10)$$

with the initial condition

$$B_t^1(k) = k - th \max_{\forall n} P(n \mid O_1^t) \qquad (11)$$

where O_u^v denotes the observation sequence $\{o_u, o_{u+1}, \ldots, o_v\}$, and $A_t^\ell(k)$ records the corresponding node with $B_t^\ell(k)$. Finally, $A_T^\ell(k)$ represents the k-th best termination node for all ℓ-character words. Sorting these $A_T^\ell(\cdot)$ for all possible ℓ's by the associated $B_T^\ell(\cdot)$ will rank the choices for all possible strings. The details can be found in [4].

4 Experiments

The training character images are extracted from the 3,103 training word images (USPS Database-IV) while the testing character images are extracted from another 1,034 word images. Totally, 20,5129 character images are used for training while another 6,609 character images are used for testing. After re-estimation, smoothing and normalization, up to 80% character recognition accuracy is achieved.

The proposed systems are further evaluated using the 3,000 test word images. For each test word image, two dictionaries are randomly generated with the size of 100 and 1000 words. From Table 1, we find that the MD-HMM approach performs much better than PD-HMM but at the cost of much lower speed (especially when the dictionary is large). Also, the performance of the new HWR system is comparable to our previous systems [1,2] when the top few choices are considered. Moreover, we have found that the PD-HMM version of MLHMM has obtained near 95% recognition rate at the top 30 choices even with a 1000-word dictionary. Thus, the PD-HMM version of MLHMM could be a good frontier word filter, i.e., the PDHMM based system is first used to pick the most likely words and to shrink the dictionary. The MDHMM based based system is then used to reorder the recognition results for better accuracy. This technique overcomes the slow speed of MDHMM based system. The prototype systems described here are promising and there remains a lot of room for improvement in terms of early use of the dictionary and more judicious training.

References

[1] M.-Y. Chen, A. Kundu, and J. Zhou, "Off-line handwritten word recognition using a hidden Markov model type stochastic network," *IEEE Trans. Pattern Anal., Machine Intell.*, vol. 16, pp. 481–496, May 1994.

[2] M.-Y. Chen, A. Kundu, and S. N. Srihari, "Unconstrained handwritten word recognition using continuous density variable duration hidden Markov model," in *Proc. IEEE Int. Conference on Acoust., Speech, Signal Processing*, vol. 5, (Minneapolis, Minnesota), pp. 105–108, April 1993. Accepted in IEEE Trans. on Image Processing for Publication.

[3] L. R. Rabiner, "A tutorial on hidden Markov model and selected applications in speech recognition," *IEEE Proceedings*, vol. 77, pp. 257–286, Feb. 1989.

[4] M.-Y. Chen, *Handwritten Word Recognition Using Hidden Markov Models.* PhD thesis, State University of New York at Buffalo, August 1993.

[5] K.-F. Lee, H.-W. Hon, and R. Reddy, "An overview of the SPHINX speech recognition system," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 38, pp. 35–45, Jan. 1990.

100-Word Dictionary				
HMM Strategy	Top 1	Top 5	Top 20	Top 30
MD-HMM	67.0%	86.6%	96.1%	–.-%
PD-HMM	51.1%	71.3%	86.2%	94.7%

1000-Word Dictionary				
HMM Strategy	Top 1	Top 5	Top 30	Top 100
MD-HMM	46.1%	69.2%	84.7%	–.-%
PD-HMM	20.6%	42.6%	65.8%	94.5%

Table 1: The systems are evaluated using 3,000 postal word images with two different dictionaries.

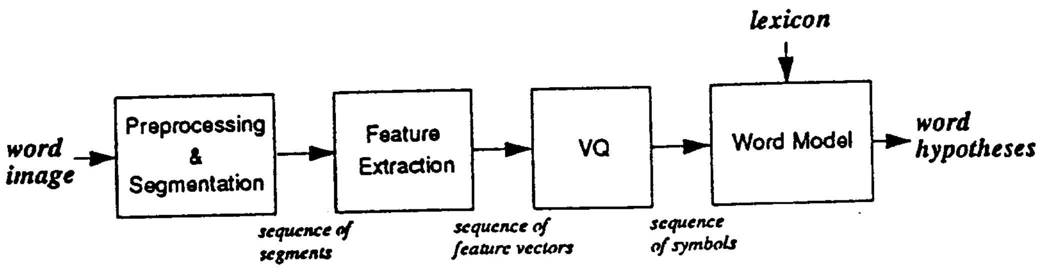

Figure 1: The system diagram for the MLHMM based HWR system.

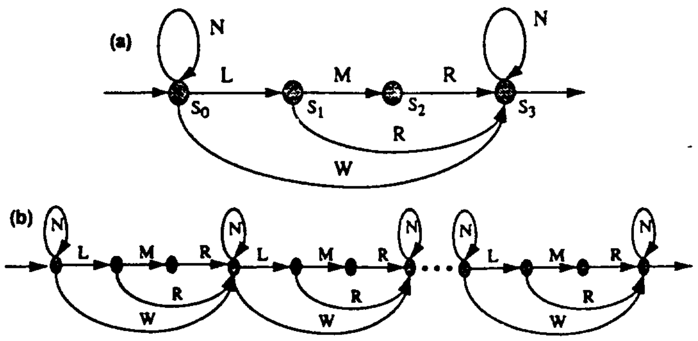

Figure 2: (a) A character is modeled as an HMM with 4 states, 5 observation symbols and 7 transitions. (b) For the MD-HMM approach, the word models are built by cascading the character models while eliminating one 'N' symbols between characters.