A Stochastic Model Combining Discrete Symbols and Continuous Attributes and Its Application to Handwriting Recognition

Hanhong Xue and Venu Govindaraju

CEDAR, Department of Computer Science and Engineering
SUNY at Buffalo, Buffalo, NY 14260, USA
{hxue, govind}@cedar.buffalo.edu

Abstract. This paper introduces a new stochastic framework of modeling sequences of features that are combinations of discrete symbols and continuous attributes. Unlike traditional hidden Markov models, the new model emits observations on transitions instead of states. In this framework, a feature is first labeled with a symbol and then a set of feature-dependent continuous attributes is associated to give more details of the feature. This two-level hierarchy is modeled by symbol observation probabilities which are discrete and attribute observation probabilities which are continuous. The model is rigorously defined and the algorithms for its training and decoding are presented. This framework has been applied to off-line handwritten word recognition using high-level structural features and proves its effectiveness in experiments.

1 Introduction

Stochastic models, especially hidden Markov models (HMMs), have been successfully applied to the field of off-line handwriting recognition in recent years. These models can generally be categorized as being either discrete or continuous, depending on their observation types.

Bunke *et al.* [1] model an edge in the skeleton of a word image by its spatial location, degree, curvature and other details, and derived 28 symbols by vector quantization for discrete HMMs. Chen *et al.* [2] use 35 continuous features including momental, geometrical, topological and zonal feature in building continuous density and variable duration HMMs. Mohammed and Gader [3] incorporate locations of vertical background-foreground transitions in their continuous density HMMs. Senior and Robinson [4] describe a discrete HMM system modeling features extracted from a grid. The features include information such as the quantized angle that a stroke enters from one cell to another and the presence of dots, junctions, endpoints, turning points and loops in a cell. El-Yacoubi *et al.* [5] adopt two sets of discrete features, one being global features (loops, ascenders, descenders, etc.) and the other being bidimensional dominant transition numbers, in their HMMs.

As can be seen, most of the previous studied stochastic models focus on modeling low-level statistical features and fall into being either discrete or continuous.

D. Lopresti, J. Hu, and R. Kashi (Eds.): DAS 2002, LNCS 2423, pp. 70–81, 2002.

In studying handwriting recognition using high-level structural features, such as loops, crosses, cusps and arcs shown in Figure 1(a), we find it more accurate to associate these features, which are discrete symbols, with some continuous attributes. These attributes include position, orientation, and angle between strokes as shown in Figure 1(b) and they are important to recognition tasks because more details are given regarding the feature. For example, vertical position is critical in distinguishing an 'e' and an 'l' when both of them are written in loops. Since the vertical position can be anywhere in the writing zone, it takes continuous values.

Therefore, this paper tries to explore a new approach of modeling sequences consisting of discrete symbols and their continuous attributes for off-line handwriting recognition.

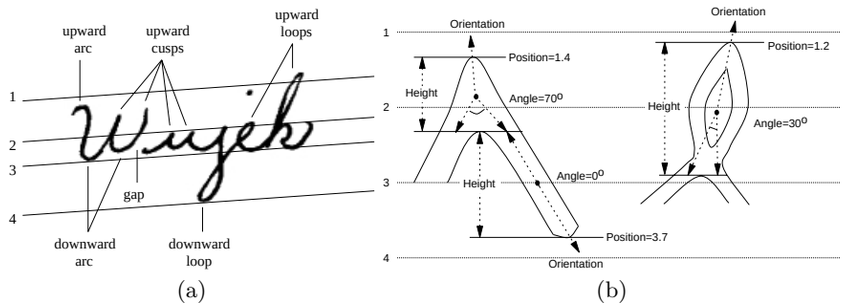

Fig. 1. High-level structural features and their possible continuous attributes

Table 1. Example of structural features and their attributes, extracted from Figure 1(a)

character	symbol	position	orientation	angle
W	upward arc	1.2		126^o
	downward arc	3.1		143^o
	upward cusp	1.6	74^o	
	downward arc	2.9		153^o
	upward cusp	1.4	82^o	
	gap	0.2		
...	...			
k	downward cusp	3.0	-90^o	
	upward loop	1.0		
	downward arc	3.0		149^o
	upward cusp	2.0	80^o	

2 Structural Features

Table 2 lists the structural features that are used to model handwriting in this paper. In these features, long cusps and short cusps are separated by thresholding their vertical length. Left-terminated arcs are arcs whose stroke ends at its left side; right-terminated arcs are arcs whose stroke ends at its right side. All other features can be easily understood. For each feature, there is a set of continuous attributes associated with it. (Refer to Figure 1 for the meaning of the attributes.) Position is relative to reference lines. Orientation and angle are in radius (shown in degrees in Figure 1 to be better understood). Width is relative to average character width. All the features and their attributes are obtained by the skeletal graph approach described in [6].

To model the distribution of structural features, we need to also consider their attributes. Suppose the full description of a structural feature is given as (u, v) where u is the feature category, such as any of the 16 listed in Table 2, and v is a vector of attributes associated with the category. So the probability of having (u, v) can be decomposed into two parts:

$$P(u, v) = P(u)P(v|u), \tag{1}$$

where the distribution of $P(u)$ is discrete and that of $P(v|u)$ is continuous. Therefore, $P(u)$ can be modeled by discrete probabilities and $P(v|u)$ can be modeled by multivariate Gaussian distributions. The advantage of such a decomposition is that each feature category can have different number of attributes.

Table 2. Structural features and their attributes. 16 features in total. Attributes associated with a feature are marked.

structural feature	position	orientation	angle	width
upward loop	X			
upward long cusp	X	X		
upward short cusp	X	X		
upward arc	X		X	
upward left-terminated arc	X		X	
upward right-terminated arc	X		X	
circle	X			
downward loop	X			
downward long cusp	X	X		
downward short cusp	X	X		
downward arc	X		X	
downward left-terminated arc	X		X	
downward right-terminated arc	X		X	
cross	X			
bar	X			
gap				X

3 Model Definition

To model sequences of structural features with continuous attributes, we define stochastic finite-state automaton $\lambda =< S, L, A >$ as follows.

- $S = \{s_1, s_2, ..., s_N\}$ is a set of states, assuming single starting state s_1 and single accepting state s_N.
- $L = \{l_1, l_2, ..., l_M\}$ is a set of discrete symbols corresponding to feature categories. For each symbol, there is a set of continuous attributes to describe its details. So an observation is represented as $o = (u, v)$ where $u \in L$ is a symbol and v a vector of continuous values. A special symbol, the null symbol ϵ, has no attributes and does not appear in the input.
- $A = \{a_{ij}(o)\}$, the *observation probability*, is a set of probability density functions (pdfs), where $a_{ij}(o)$ is the pdf of features observed while transitioning from state i to state j. The sum of outgoing probabilities from a state must be 1, *i.e.*

$$\sum_j [a_{ij}(\epsilon) + \sum_u \int_v a_{ij}(u, v)dv] = 1 \tag{2}$$

  for all state i.

Given a non-null observation $o = (u, v) = (l_k, v)$, the observation probability is decomposed into two parts:

$$a_{ij}(o) = P(l_k, v|i, j) = P(l_k|i, j)P(v|l_k, i, j) = f_{ij}(l_k)g_{ijk}(v). \tag{3}$$

The first part is called the *symbol observation probability*, which is the probability of observing a symbol l_k regardless its attributes. The second part is called the *attribute observation probability*, which is defined by a probability density function on the attributes that the symbol l_k has. The null symbol does not have any attribute, so its observation probability is denoted as

$$a_{ij}(\epsilon) = f_{ij}(\epsilon), \tag{4}$$

where only the symbol observation probability presents. Unlike in HMMs, here we do not have pure transition probabilities since observations are actually emitted by transitions instead of states.

We model attribute observation probabilities by multivariate Gaussian distributions

$$g_{ijk}(v) = \frac{1}{\sqrt{(2\pi)^{d_k}|\sigma_{ijk}|}} e^{-\frac{1}{2}[(v-\mu_{ijk})'\sigma_{ijk}^{-1}(v-\mu_{ijk})]}, \tag{5}$$

where μ_{ijk} is the average of attributes of symbol l_k on the transition from state i to state j, σ_{ijk} is the covariance matrix of these attributes, and d_k is the number of attributes symbol l_k has. In practice, we assume the covariance matrix is diagonal for simplicity and for the fact that attributes involved are strongly independent to each other. It should be noticed that symbols are not required to have the same number of attributes. As the number of attributes increases,

observation probabilities decrease exponentially. Therefore, they are actually normalized by taking their d_k-th root to make them comparable.

The input to a model is an observation sequence $O = (o_1, o_2, ..., o_T)$ where $o_t = (u_t, v_t)$, $u_t \in L$ and v_t is a vector of continuous values. For example, $u_1 =$ "upward arc", $v_1 = (1.2, 126^o)$ and $u_6 =$ "gap", $v_6 = (0.2)$ in Table 1.

We define the predicate $Q(t, i)$ to mean that the model is in state i at time t. Given the input, a state sequence $Q(t_0, q_0), Q(t_1, q_1), ..., Q(t_W, q_W)$ describes how the model interprets the input by transitioning from the starting state at time 0 to the accepting state at time T. So it is required that $t_0 = 0$, $q_0 = 1$, $t_W = T$ and $q_W = N$.

In this stochastic model, the general problem is to decide observation probabilities which also imply the model topology. At the training phase, the Forward-Backward algorithm can be used to decide observation probabilities given a set of sample observation sequences; while at the decoding phase, the Viterbi algorithm gives a good approximation to the probability of having some input given the model. Details will be given in later sections.

4 Training

The training is done by the Forward-Backward or Baum-Welch algorithm [7], with a little modification. This algorithm is a subcase of the Expectation-Maximization algorithm, which guarantees to converge to a local extremum.

4.1 Forward and Backward Probabilities

The forward probability $\alpha_j(t) = P(o_1, o_2, ...o_t, Q(t, j)|\lambda)$ is defined as the probability of being in state j after the first t observations given the model. It can be recursively calculated by the following equation.

$$\alpha_j(t) = \begin{cases} 1 & , j = 1, t = 0 \\ \sum_i (\alpha_i(t) a_{ij}(\epsilon) + \alpha_i(t-1) a_{ij}(o_t)) & , \text{otherwise} \end{cases} \tag{6}$$

The first term in the sum accounts for observing the null symbol, which does not consume any input observation, and the second term accounts for observing some non-null symbol in the input.

The backward probability $\beta_i(t) = P(o_{t+1}, o_{t+2}, ...o_T, Q(t, i)|\lambda)$ is defined as the probability of being in state i before the last $T - t$ observations given the model. It can be calculated recursively as follows.

$$\beta_i(t) = \begin{cases} 1 & , i = N, t = T \\ \sum_j (a_{ij}(\epsilon)\beta_j(t) + a_{ij}(o_t)\beta_j(t+1))s & , \text{otherwise} \end{cases} \tag{7}$$

Similarly, the two terms in the sum account for the null symbol and some non-null symbol in the input, respectively.

Finally, $\alpha_N(T) = \beta_1(0) = P(O|\lambda)$ is the overall probability of having the input given the model.

4.2 Re-estimation

Define $\omega_{ij}(t) = P(Q(t,i), Q(t,j)|O,\lambda)$ as the probability of observing ϵ while transitioning from state i to state j at time t, and $\tau_{ij}(t) = P(Q(t-1,i), Q(t,j)|O,\lambda)$ as the probability of observing a non-null symbol while transitioning from state i at time $t-1$ to state j at time t. $\omega_{ij}(t)$ and $\tau_{ij}(t)$ can be computed by the following equations.

$$\begin{aligned}\omega_{ij}(t) &= P(Q(t,i), Q(t,j)|O,\lambda) \\ &= \frac{P(Q(t,i),Q(t,j),O|\lambda)}{P(O|\lambda)} \\ &= \frac{P(o_1,o_2,\ldots o_t,Q(t,i)|\lambda)a_{ij}(\epsilon)P(Q(t,j),o_{t+1},o_{t+2},\ldots o_T|\lambda)}{P(O|\lambda)} \\ &= \frac{\alpha_i(t)a_{ij}(\epsilon)\beta_j(t)}{\alpha_N(T)}\end{aligned} \tag{8}$$

$$\begin{aligned}\tau_{ij}(t) &= P(Q(t-1,i), Q(t,j)|O,\lambda) \\ &= \frac{P(Q(t-1,i),Q(t,j),O|\lambda)}{P(O|\lambda)} \\ &= \frac{P(o_1,o_2,\ldots o_{t-1},Q(t-1,i)|\lambda)a_{ij}(o_t)P(Q(t,j),o_{t+1},o_{t+2},\ldots o_T|\lambda)}{P(O|\lambda)} \\ &= \frac{\alpha_i(t-1)a_{ij}(o_t)\beta_j(t)}{\alpha_N(T)}\end{aligned} \tag{9}$$

The symbol observation probability $f_{ij}(u)$ is re-estimated as the expected number of transitions from state i to state j seeing symbol u divided by the expected number of transitions out from state i.

$$\hat{f}_{ij}(u) = \begin{cases} \frac{\sum_t \omega_{ij}(t)}{\sum_j \sum_t (\omega_{ij}(t)+\tau_{ij}(t))}, u = \epsilon \\ \frac{\sum_{t,u_t=u} \tau_{ij}(t)}{\sum_j \sum_t (\omega_{ij}(t)+\tau_{ij}(t))}, u \neq \epsilon \end{cases} \tag{10}$$

This estimation directly conforms to the constraint that the sum of outgoing probabilities from a state must be 1.

Since the null symbol does not have any attribute, re-estimation of attribute observation probability is only necessary for non-null symbols. The definition of attribute observation probability has two parameters. The average of attributes of symbol l_k on the transition from state i to state j is re-estimated as

$$\hat{\mu}_{ijk} = \frac{\sum_{t,u_t=l_k} \tau_{ij}(t) v_t}{\sum_{t,u_t=l_k} \tau_{ij}(t)} \tag{11}$$

and the covariance of these attributes is similarly re-estimated as

$$\hat{\sigma}_{ijk} = \frac{\sum_{t,u_t=l_k} \tau_{ij}(t)(v_t - \mu_{ijk})'(v_t - \mu_{ijk})}{\sum_{t,u_t=l_k} \tau_{ij}(t)}. \tag{12}$$

Notice that the denominators in the above two equations are the same as the numerator of the $u \neq \epsilon$ case in Equation 10.

4.3 Parameter Tying

Sometimes model parameters cannot be reliably re-estimated due to large variations or the lack of sufficient samples. For example, self-transitions absorb extra features that are more likely to have all kinds of attributes, so their parameters tend to be less reliable. In this case, parameters for all self-transitions in a model can be tied in re-estimation and shared in decoding.

We tie the attribute observation probabilities for all self-transitions in a model. Let μ_k and σ_k be the mean and the variance of the attributes of l_k on all self-transitions, respectively. They are re-estimated by the following equations.

$$\hat{\mu}_k = \frac{\sum_i \sum_{t,u_t=l_k} \tau_{ii}(t) v_t}{\sum_i \sum_{t,u_t=l_k} \tau_{ii}(t)}, \tag{13}$$

$$\hat{\sigma}_k = \frac{\sum_i \sum_{t,u_t=l_k} \tau_{ii}(t)(v_t - \mu_k)'(v_t - \mu_k)}{\sum_i \sum_{t,u_t=l_k} \tau_{ii}(t)}. \tag{14}$$

5 Decoding

The decoding is done by the Viterbi algorithm, which produces the most probable state sequence for a given input O. Define $\gamma_i(t)$, the Viterbi probability, as the highest probability of being in state i at time t produced by one state sequence, then it can be recursively calculated as follows.

$$\gamma_j(t) = \begin{cases} 1 & , j = 1, t = 0 \\ \max(\max_i \ \gamma_i(t) a_{ij}(\epsilon), \ \max_i \ \gamma_i(t-1) a_{ij}(o_t)) & , \text{otherwise} \end{cases} \tag{15}$$

Finally, $\gamma_N(T)$ is the Viterbi probability of observing the entire sequence O given the model.

6 Modeling Words

Word models are obtained by concatenating character models. However, word modeling is different for training and decoding, as shown in Figure 2. During training, image truths are provided with the case (uppercase or lowercase) of all letters determined, so linear concatenation is sufficient. In decoding, since the image truth is unknown, the model of a candidate word must allow all possible combinations of cases. Therefore, bi-gram case probabilities, which are the probabilities of having the case of a character given the case of its previous character, are applied to modeling the case change between neighboring letters.

We obtain the bi-gram case probabilities from training data and give them in Table 3. According to this table, it is much more probable to have uppercase letters as the first letter of a word than have lowercase letters. This is because our training set is made of postal words that are usually capitalized. It also can be seen, a letter is likely to have the same case as its previous letter, with exceptions for vowels that are more likely to be in lowercase than in uppercase.

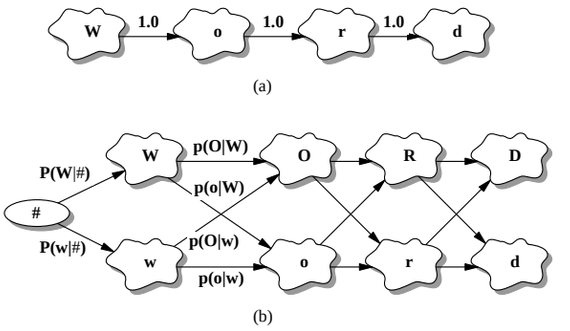

Fig. 2. Connecting character models to build word models for (a) training, and (b) decoding.

Table 3. Probabilities of the case of a character given the case of its previous character. If a character begins a word, then its previous character is #.

	a	A	b	B	c	C	d	D	e	E	f	F	g	G
#	0.308	0.692	0.002	0.998	0.022	0.978	0.015	0.985	0.011	0.989	0.029	0.971	0.073	0.927
lowercase	0.982	0.018	0.982	0.018	0.989	0.011	0.992	0.008	0.993	0.007	0.987	0.013	0.998	0.002
uppercase	0.644	0.356	0.065	0.935	0.145	0.855	0.290	0.710	0.660	0.340	0.339	0.661	0.267	0.733

	h	H	i	I	j	J	k	K	l	L	m	M	n	N
#	0.010	0.990	0.021	0.979	0.047	0.953	0.008	0.992	0.015	0.985	0.065	0.935	0.103	0.897
lowercase	0.992	0.008	0.997	0.003	0.500	0.500	0.966	0.034	0.997	0.003	0.989	0.011	0.981	0.019
uppercase	0.675	0.325	0.748	0.252	0.500	0.500	0.172	0.828	0.489	0.511	0.588	0.412	0.228	0.772

	o	O	p	P	q	Q	r	R	s	S	t	T	u	U
#	0.046	0.954	0.029	0.971	0.333	0.667	0.006	0.994	0.029	0.971	0.022	0.978	0.111	0.889
lowercase	0.998	0.002	0.972	0.028	0.947	0.053	0.982	0.018	0.984	0.016	0.993	0.007	0.998	0.002
uppercase	0.666	0.334	0.451	0.549	0.200	0.800	0.517	0.483	0.176	0.824	0.324	0.676	0.839	0.161

	v	V	w	W	x	X	y	Y	z	Z
#	0.018	0.982	0.025	0.975	0.500	0.500	0.099	0.901	0.500	0.500
lowercase	0.993	0.007	0.993	0.007	0.958	0.042	0.987	0.013	0.950	0.050
uppercase	0.178	0.822	0.076	0.924	0.250	0.750	0.436	0.564	0.500	0.500

7 Experimental Results

We implement the above-described stochastic models for handwritten word recognition. Figure 3 pictures the control flow of the system. Details of the entire training-decoding process are given as follows.

1. Feature sequences of training characters, training words and testing words are extracted.
2. Character models, including both uppercase and lowercase, are built from training feature sequences extracted on character images. The number of states in a model is simply decided according to the average length of the training sequences and a state i is connected to a state j if (a) $j = i$, or, (b) $j > i$ and $j - i \equiv 1 \bmod 2$. Therefore, the models are guaranteed to be acyclic in topology (except for self transitions) and the connections are not fully

dense. During training, attribute observation probabilities on self transitions will be tied for all states because these transitions absorb excessive features that have large attribute variations. Table 4 gives the number of states for each character model.

3. The models are trained on character images. [1] To prevent over-training, we prune the model to allow only transitions with symbol observation probabilities above a threshold (0.001) and re-assign an attribute-dependent minimum variance to any variance smaller than it.
4. The models are trained on word images, with gaps between characters considered by a trailing transition behind each character model.
5. Uppercase and lowercase character models are interconnected by bi-gram probabilities to get word models for matching against an input feature sequence.

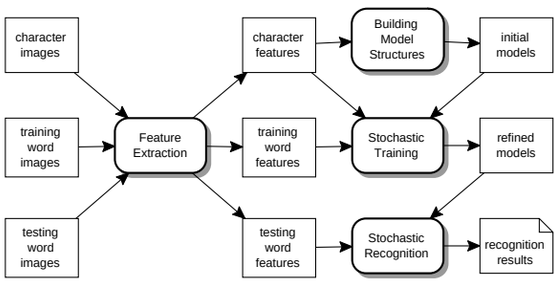

Fig. 3. Control flow of the word recognition system, including both training and decoding(recognition)

Table 4. Numbers of states in character models. (8.0 on average for uppercase and 8.4 on average for lowercase)

character	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
# states	8	8	7	7	8	8	9	8	7	8	8	8	11	9	7	7	7	8	7	8	8	8	10	7	9	8
character	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
# states	8	9	8	8	8	8	8	9	8	8	8	8	11	9	7	8	9	8	7	8	9	9	11	8	9	8

In order to test the effectiveness of associating continuous attributes with discrete symbols, we start without any attributes and add in them one by one. The first attribute added is the width of gaps and the position of all other

[1] It is possible to skip this step and train the models directly on word images. However, this step gives a chance to reach a better local extremum for the next step according to our experimental experiences.

structures. The second attribute added is the orientation of cusps and the angle of arcs. It should be noticed that some features, such as gaps and loops, do not have more than one attribute, so eventually we are modeling features with different numbers of attributes. Table 5 shows accuracy rates obtained on a set of 3,000 US postal images (CEDAR BHA testing set) with lexicons of different sizes. This testing set is considered as relatively difficult because some words that are very similar to the truth have been inserted in the lexicon to confuse recognizers. It can be seen that the addition of continuous attributes significantly improves the recognizer's performance, especially when the lexicon size is large.

Table 5. Recognition results using different number of continuous attributes, on lexicon of size 10, 100, 1000 and 20000.

Lexicon size	10			100			1000			20000		
max # attr.	0	1	2	0	1	2	0	1	2	0	1	2
Top 1	94.46	95.66	96.56	80.14	85.15	89.12	62.56	69.97	75.38	38.35	50.40	58.14
Top 2	97.96	98.19	98.77	88.28	91.56	94.06	74.87	82.68	86.29	48.10	49.15	66.49
Top 10				96.90	97.93	98.19	88.19	92.29	94.39	66.76	75.63	81.31
Top 20							91.99	94.59	96.50	73.60	80.41	85.71
Top 100										86.45	89.79	93.39

Table 6 compares the stochastic recognizers against other recognizers tested on the same data set. The first one is a recognizer modeling image segments by continuous density variable duration HMMs [8]. The second one is an approach of over-segmentation followed by dynamic programming on segment combinations [9]. The third one is a recently improved version of the second one by incorporating Gaussian mixtures to model character clusters [10]. To compare, the stochastic recognizer is better than [8] and [9] but worse than [10]. This is largely due to the inconsistency in the feature extraction procedure where many different heuristics are used to identify structural features and to arrange them approximately in the same order as they are written. For some images, the procedure produces unexpected feature sequences, such as features in reversed order, which are not familiar to the trained models and cause recognition errors.

8 Conclusions and Future Work

This paper presents a stochastic framework of modeling features that consist of discrete symbols associated with continuous attributes, aiming at its applications to off-line handwritten word recognition using high-level structural features. In this framework, different sets of attributes can be associated with different discrete symbols, providing variety and flexibility in modeling details. As supported by experiments, the addition of continuous attributes to discrete symbols does improve the overall recognition accuracy significantly.

Now we are investigating some possible ways of improving the recognizer. The first one is to expand the feature set to capture more detailed handwriting

Table 6. Performance comparison against other three word recognizers on the same testing set.

Lex size		[8]	[9]	[10]	This paper
10	Top 1	93.2	96.80	96.86	96.56
	Top 2		98.63	98.80	98.77
100	Top 1	80.6	88.23	91.36	89.12
	Top 2		93.36	95.30	94.06
	Top 3	90.2			
	Top 20		98.93	99.07	99.10
1000	Top 1	63.0	73.80	79.58	75.38
	Top 2		83.20	88.29	86.29
	Top 3	79.3			
	Top 5	83.9		93.29	91.69
	Top 50		98.70	98.00	98.40
20000	Top 1			62.43	58.14
	Top 2			71.07	66.49
	Top 10			83.62	81.31
	Top 20			87.49	85.71
	Top 100			93.59	93.39

styles, which means more symbols and more attributes. The second one is to make feature extraction more robust to writing styles and image quality. The third one is to optimize the model topology by learning from training examples. All these remain challenging tasks for the future.

References

1. H. Bunke, M. Roth, and E. Schukat-Talamazzini, "Off-line cursive handwriting recognition using hidden Markov models," *Pattern Recognition*, vol. 28, no. 9, pp. 1399–1413, 1995.
2. M. Chen, A. Kundu, and S. Srihari, "Variable duration hidden Markov model and morphological segmentation for handwritten word recognition," *IEEE Transactions on Image Processing*, vol. 4, pp. 1675–1688, December 1995.
3. M. Mohammed and P. Gader, "Handwritten word recognition using segmentation-free hidden Markov modeling and segmentation-based dynamic programming techniques," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, pp. 548–554, May 1996.
4. A. Senior and A. Robinson, "An off-line cursive handwriting recognition system," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 3, pp. 309–321, 1998.
5. A. El-Yacoubi, M. Gilloux, R. Sabourin, and C. Y. Suen, "An HMM-based approach for off-line unconstrained handwritten word modeling and recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, pp. 752–760, August 1999.
6. H. Xue and V. Govindaraju, "Building skeletal graphs for structural feature extraction on handwriting images," in *International Conference on Document Analysis and Recognition*, (Seattle, Washington), pp. 96–100, September 2001.

7. L. Baum, "An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes," *Inequalities*, vol. 3, pp. 1–8, 1972.
8. M. Chen, *Handwritten Word Recognition Using Hidden Markov Models*. PhD thesis, State University of New York at Buffalo, September 1993.
9. G. Kim and V. Govindaraju, "A lexicon driven approach to handwritten word recognition for real-time applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, pp. 366–379, April 1997.
10. S. Tulyakov and V. Govindaraju, "Probabilistic model for segmentation based word recognition with lexicon," in *Proceedings of Sixth International Conference on Document Analysis and Recognition*, (Seattle), pp. 164–167, September 2001.