REAL-TIME ON-LINE UNCONSTRAINED HANDWRITING RECOGNITION USING STATISTICAL METHODS

Krishna S. Nathan, Homayoon S. M. Beigi, Jayashree Subrahmonia, Gregory J. Clary and Hiroshi Maruyama

Handwriting Recognition Group
IBM Research, T.J. Watson Center, Yorktown Heights, NY 10598

ABSTRACT

We address the problem of automatic recognition of unconstrained handwritten text. Statistical methods, such as hidden Markov models (HMMs) have been used successfully for speech recognition and they have recently been applied to the problem of handwriting recognition as well. In this paper, we will discuss a general recognition system for large vocabulary, writer independent, unconstrained handwritten text. "Unconstrained" implies that the user may write in any style e.g. printed, cursive or in any combination of styles. This is more representative of typical handwritten text where one seldom encounters purely printed or purely cursive forms. Furthermore, a key characteristic of the system described in this paper is that it performs recognition in *real-time* on 486 class PC platforms without the large amounts of memory required for traditional HMM based systems. We focus mainly on the writer independent task. Some initial writer dependent results are also reported. An error rate of 18.9% is achieved for a writer-independent 21,000 word vocabulary task in the absence of any language models.

1. INTRODUCTION

The automatic recognition of unconstrained on-line handwritten text is addressed. In on-line recognition data are collected on an electronic tablet that traces the movement of the pen, thereby preserving temporal information. Several approaches to the problem have been reported in the literature; e.g. [7] use a TDNN for the recognition of discrete characters. More recently, statistical methods, such as hidden Markov models (HMMs) that have been used successfully for speech recognition [1], have been applied to the problem of automatic handwriting recognition as well [6, 8, 5, 4]. In [6] continuous parameter HMMs are used to recognize characters written in isolation. [8] perform recognition of writer dependent, purely cursive handwriting using a system developed for speech recognition. This paper describes a HMM based system that is not restricted to a particular style, i.e., purely cursive or purely discrete, of writing. The writing may be any combination of the 2 styles, a situation frequently encountered in practice. Much effort was spent in designing a system that could run in real time on small PC platforms with limited memory. Compromises were made in order to reduce computational and memory requirements sometimes at the expense of accuracy. Section 2 describes the various components of the recognizer. The recognition is carried out in two steps. In the first step a simpler model is used to generate a short list of potential candidate strings. This is referred to as the fast match (FM). Subsequently, a computationally more expensive model is used to reorder each word in the short list. Section 3 discusses writer independent results for different vocabulary sizes. Some writer dependent results are also presented. The main purpose of this paper is to report on the shape matching capability of the model. Consequently, experiments with language models or grammars are not described.

2. SYSTEM DESCRIPTION

Figure 1 is an overall block diagram of the system.

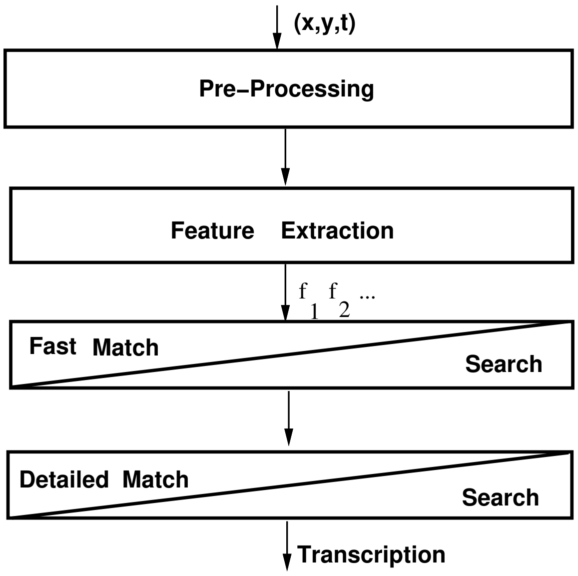

Figure 1: Block Diagram of Recognition System

2.1. Pre-processing and Feature Extraction

The data are collected as a stream of (x, y) points indexed in time, sampled at rates between 70Hz and 100Hz. The incoming temporally equi-spaced points are normalized to a standard size and re-sampled spatially [2]. This removes inconsistencies due to velocity and makes recognition more robust to the different writing speeds inherent to writer independent samples. Similarly, the size normalization, although not crucial for writer dependent recognition, was found to be essential for the writer independent case. A feature vector consisting of $\Delta x, \Delta y, cos\theta$ and $sin\theta$ is constructed at equi-spaced points along the trajectory. θ is defined as the angle at the sample. Contextual information is incorporated by splicing several individual feature vectors into one large feature vector such that it spans a window of adjacent points. The window centers are typically located at local extrema in x and y. Each such feature vector is then projected onto a lower dimensional space. We will refer to this vector as a frame. A typical word may be represented by 30 such frames, each representing a neighborhood of points along the trajectory.

2.2. Fast Match

In order to limit computation we use a degenerate single state model to generate a short list of candidate hypotheses [3]. Figure 2 shows the topology for the single state model. The feature extraction yields a sequence of frames,

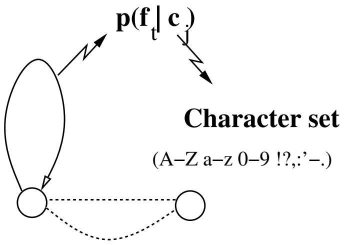

Figure 2: Single state model for a character

$f_1, f_2, \ldots$ corresponding to the handwritten text. A Gaussian mixture model determines the probability that a particular character c_j gives rise to a frame f_t viz. $p(f_t|c_j) = \sum_i p(f_t|g_i)p(g_i|c_j)$. g_i is a Gaussian with diagonal covariance matrix. The distributions g_i are obtained during the training phase by un-supervised clustering of all the frames. Also during training, the mixture coefficients $p(g_i|c_j)$ are estimated via the EM algorithm. During decoding each frame is assumed to be independent of all the others and the probability that a particular character gives rise to a set of frames is merely the product of the individual $p(f_t|c_j)$ that compose that block of frames. Note that this simplified model does not provide duration modeling. In addition, there is no distinction between the beginning of a character and the end of the character. A single output distribution characterizes the entire character. The notion of sub-models for states or sub-divisions of a character is introduced in the detailed match discussed further below.

There is no notion of segmentation prior to recognition. The segmentation arises as a natural consequence of recognition. This is done by means of the search described below.

2.3. Detailed Match

The fast match described above can be shown to be equivalent to a single state HMM; there is no notion of relative position of frames within a character. In the detailed match model this state degeneracy is removed and each character is modeled by a series of states, each of which has associated with it an output distribution corresponding to the portion of the character that it models.

Figure 3 shows the HMM for character i. The HMM

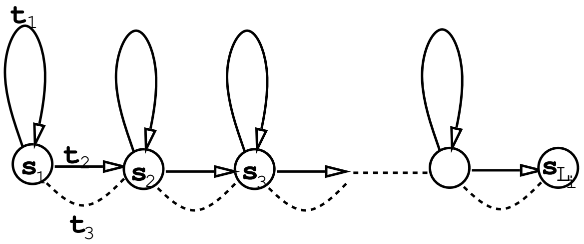

Figure 3: HMM Topology

has L_i states labeled $s_1, s_2, \ldots, s_{L_i}$, where L_i is the average number of frames for character i. Associated with each of the L_i states, $s_1, s_2, \ldots, s_{L_i}$, is a set of three transitions labeled $t_1, t_2 and t_3$. Transitions t_1 and t_2 result in the emission of an observation feature vector. The number of states L_i, the state transition probabilities, $p(s_i, t_j)$ and the output probability distributions $p(f_t \mid s_i, t_j)$ completely specify the model. The transitions t_1 and t_2 for a given state are tied, i.e., $p(f_t \mid s_i, t_1) = p(f_t \mid s_i, t_2)$. Hence the output distribution is associated with the state alone and can be written as $p(f_t \mid s_i)$. The output probabilities are determined from a mixture of tied Gaussian distributions, and can be written as : $p(f_t \mid s_i) = \sum_k p(f_t \mid g_k)p(g_k \mid s_i)$, where the summation is over the entire pool of distributions. Hence, the HMM is completely specified by the state transition probabilities, $p(s_i, t_j)$, and the mixture coefficients, $p(g_k \mid s_i)$.

The HMMs are initialized from the fast match models by replicating the single state model L_i times. The HMMs are trained using characters written in isolation and words written in an unconstrained manner using either Viterbi or forward-backward training. In our system, there is no significant difference in accuracy between the two training schemes.

In decoding, the probability of a set of frames given a character i is given by the probability of the most probable state sequence that could generate that set of frames. The optimal sequence of characters that make up the word are determined by a time synchronous beam search.

2.4. Beam Search

The 20,000+ word lexicon is stored in a structure that merges common prefixes and suffixes. Since the search is lexicon driven, only those paths corresponding to valid words in the vocabulary are expanded. Associated with each frame is a stack containing all possible partial paths ending at that frame. The stacks are sorted by probability.

Naturally, thresholds are used to prune out low probability elements. The top element of the final stack corresponds to the recognized string. The search space is made more tractable by the introduction of constraints that limit the number of nodes that are expanded. One such constraint is a length distribution for each character. These length histograms, specifying the range of the number of frames that a given character may span, are generated during training. These distributions are used in the fast match stage since the single state model does not have the capacity to model duration or the number of frames. In the context of the multi-state model the external length distribution is not used since the length and state transition probabilities of individual models define duration.

Delayed strokes pose a problem when using left to right HMMs to model characters. Examples of delayed strokes are the dots in the characters "i" and "j", and crosses in the characters "x" and "t". These are strokes that are often temporally separated from the body of the character. In most cases, these are added after the whole word is written. The data points for these characters are not necessarily contiguous in time, thus posing a problem when using left to right models for these characters. Our solution to this problem is to train the HMM on only the non delayed strokes for these characters. Delayed strokes are stripped off before training the HMM. In decoding, the search mechanism first expands the non delayed strokes based on the frame probabilities and the character models. The delayed strokes are then incorporated based on their position relative to the non delayed strokes and their fast match probabilities.

3. EXPERIMENTS AND DISCUSSION

3.1. Data Sets

Since we were primarily interested in the writer independent performance of the recognizer, our first task was to collect data from a sufficiently large pool of writers. Approximately 100,000 characters of data were collected from a pool of 100 writers. The training set consisted of words chosen from a 20,000+ word lexicon and discrete characters written in isolation. The subjects were asked to write in their natural style and encouraged to write on a horizontal line. *No* other instructions or directions pertaining to writing style were given. The test set was composed of data collected from a separate set of 25 writers and consisted uniquely of words chosen at *random* from the same lexicon. Once again, there were no constraints on how the test data were to be written. Both native and non-native writers were included in both test and training sets. The data were collected on convertible IBM pen notebook computers. As expected, the data fall into three broad categories: purely discrete, mixed discrete and cursive, and purely cursive. Some examples are shown in Figure 4. As can be seen, there is a wide range in the 'quality' or human readability of the data. The alphabet consisted of upper and lower case characters, numbers and a few punctuation symbols and special characters.

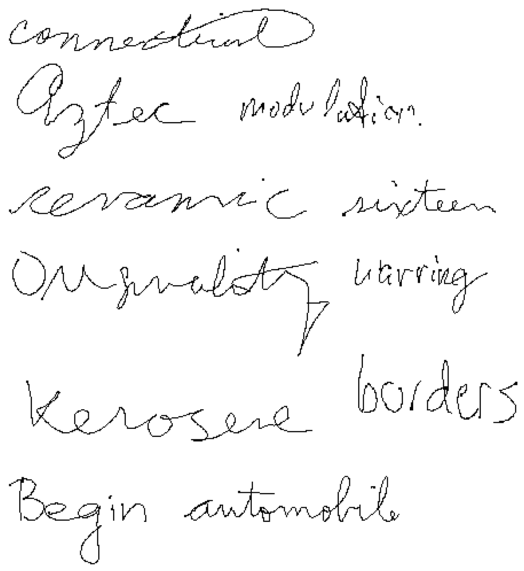

Figure 4: Sample taken from the test data

3.2. Training

In order to capture the writing styles across a broad range of writers we chose to build models not for each character but for each significant variation of each character. For example, a character may differ in the number of pen strokes, direction of pen movement or in actual shape itself. An automatic un-supervised procedure is used to identify these variations which we call lexemes. For this set, approximately 150 lexemes were generated. Individual baseforms, single and multiple state, are trained for each lexeme. Since on average each HMM consists of 6 states, this would result in 900 distinct states. In order to reduce the parameters in the system, individual states are shared across and within baseforms.

3.3. Results

In practice casual users of handwriting recognition systems are unwilling to invest the time and effort required to train user specific systems. It is also easy to envisage scenarios where training is not an option e.g. public kiosks, points of sale etc. This was our motivation for concentrating on the writer independent task. Table 1 summarizes results for various vocabulary sizes. The detailed match results were obtained by taking the top few hypotheses from the fast match and presenting them to the multiple state model. The fast match effectively acts as a pruner. Since the test words were randomly selected we do not make use of a grammar to improve recognition performance by reducing the perplexity. Hence, the perplexity of the large vocabulary task reported is over 21,000. For the small vocabulary task, we obtain a writer independent error rate of under 9%. As expected, this increases with the size of the lexicon (or perplexity). The error rate for the large vocabulary task is about 19%. We expect this number to decrease significantly if the recognizer were used in conjunction with statistical language models suited for the task. Table 2 tabulates the recognition times for the different tasks. The recognition times per word range from 0.4 sec. for the small vocabulary task to 0.48 sec. for the large vocabulary task on an IBM RS/6000 workstation platform. On standard 486 class PC platforms, we observe recognition times that are 4 to 5 times these figures which are still sufficient for real time recognition.

Table 1 : Writer Independent word error rates (No grammar)

	Small Vocab. (3K)	Medium Vocab. (12K)	Large Vocab. (21K)
Fast Match	14.9%	25.1%	27.9%
Detailed Match	9.0%	14.9%	18.9%

Table 2 : Average recognition time in seconds for one word on a IBM/RS6000

	Small Vocab. (3K)	Medium Vocab. (12K)	Large Vocab. (21K)
Fast Match	0.29	0.33	0.36
Detailed Match	0.40	0.45	0.48

In addition, we ran a small pilot writer dependent experiment. Using the writer independent system as a starting point we built models based on additional training samples from that writer. The original writer independent recognition error rate for this writer was 27% for the 21,000 vocabulary task without a grammar – significantly higher than the average writer independent rate of 19%. With the new writer dependent models the error rate decreased to slightly below 10%.

3.4. Discussion and Future Directions

The writer independent results reported above are very encouraging in light of the fact that no grammar is utilized. Our studies show that a human has an error rate of 7% when presented with the words from the above test set in the absence of any context. Applications such as note-taking (the equivalent of dictation in speech) can take advantage of statistical grammars that significantly reduce the perplexity of the task. Previous work has shown that such language models reduce the error rate by as much as a factor of 4 [8] for a similar task. As we mentioned earlier, one of the main goals of this work was to develop a system that could perform in real time on PC platforms with standard memory configurations. Consequently, we have opted for an aggressive pruning strategy that results in search errors that would otherwise be avoided. For similar computational reasons we limited the number of distributions in our models. Although we have tried to reduce the effect of these constraints on accuracy, it is easy to see that further decreases in error rate would result if they were relaxed.

We are currently working on several improvements to the model. Context dependent models have been shown to significantly improve performance both for speech and handwriting and is a natural enhancement. It is also possible to experiment with left and right handed (and perhaps even gender dependent) models. The size of the data set is another issue. We do not feel that 100,000 characters are sufficient data for successful writer independent recognition, e.g. [8] report writer dependent results trained on approximately 40,000 characters per writer using HMM techniques with writer dependent results comparable to ours. This is an area that we need to address in the future.

4. CONCLUSION

We have developed a HMM based system for the recognition of writer independent handwriting recognition. The writing style can be unconstrained, namely any mixture of printing or cursive styles. The system achieves an error rate of 19% on a 21,000 word vocabulary task with no grammar (perplexity of 21,000). When used in conjunction with statistical language models that reduce the perplexity of the task, the error rate is expected to decrease significantly. Equally importantly, the recognition is performed in real time on standard PC platforms. Further improvements in recognition performance can be expected as this latter constraint is relaxed due to the advent of faster PC's.

5. REFERENCES

[1] L. R. Bahl, F. Jelinek, and R. L. Mercer. "A Maximum Likelihood Approach to Continuous Speech Recognition". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5:179–190, March 1983.

[2] Homayoon S.M Beigi, K. Nathan, G. J. Clary, and J. Subrahmonia. "Size Normalization in OnLine Unconstrained Handwriting Recognition". In *ICIP94*, pages 169–172, 1994.

[3] E. J. Bellegarda, J. R. Bellegarda, D. Nahamoo, and Nathan K. S. "A Statistical Approach to Automatic Handwriting Recognition". *IEEE Transactions on Pattern Analysis and Machine Intelligence*. To appear in November, 1994.

[4] M. Chen and A. Kundu. "A Complement to Variable Duration Hidden Markov Model in Handwriting Recognition". In *ICIP94*, pages 174–178.

[5] R. J. Nag, K. H. Wong, and F. Fallside. "Script Recognition Usine Hidden Markov Models". In *ICASSP86*, pages 2071–2074, 1986.

[6] K. Nathan, J. R. Bellegarda, D. Nahamoo, and E. J. Bellegarda. "On-Line Handwriting Recognition Using Continuous Parameter Hidden Markov Models". In *ICASSP93*, volume 5, pages 121–124, 1993.

[7] M. Schenkel, I. Guyon, and D. Henderson. "On-Line Cursive Script Recognition Using Time Delay Neural Networks and Hidden Markov Models". In *ICASSP94*, volume 2, pages 637–640, 1994.

[8] T. Starner, J. Makhoul, R. Schwartz, and G. Chou. "On-Line Cursive Handwriting Recognition Using Speech Recognition Methods". In *ICASSP94*, volume 5, pages 125–128, 1994.