Acknowledged.

IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS, VOL. 29, NO. 2, APRIL 1999 321

EEM, shows that the new navigator guarantees a solution when the using the rule base obtained from the new training method and the EEM, shows that the new navigator guarantees a solution when the local minimum; third, its smooth changes of velocity and steering is both nearsighted and farsighted. A comparison of the navigator having to explore it beforehand or being supervised; second, it's free of local minimum; third, its smooth changes of velocity and steering angle; fourth, its planned path is close to the shortest path; and fifth, it’s both nearsighted and farsighted.

ACKNOWLEDGMENT

The 3-D real-time simulator used in this research work was developed by a number of students involved with the EXPECTATIONS project, of which the contributions of F. P. Fong and Y. L. Lee are acknowledged.

REFERENCES


1083–4419/99$10.00 © 1999 IEEE

I. INTRODUCTION

In spite of the difficulties due to variability and ambiguity of unconstrained handwriting, attempts to automatic recognition are now emerging armed with mature technologies as neural network [1], [2], hidden Markov model (HMM) [3]–[5], or other structural or mathematical modeling techniques [6]. Hybrid methods also exist where individual approaches complement each other so that even higher performance can be achieved [7]. Many researchers report promising performance with sophisticated architectures. However, they usually suffer from the weakness in their framework for modeling cursive patterns or are limited to tasks of small vocabulary. And systems are not rare that demand external segmentation in input stream, i.e., supplying a set of candidate letter boundaries prior to shape classification [8].

The primary focus of the paper is the word recognition and segmentation method which works over a network of HMM’s. In such a network an individual HMM models a letter or a ligature. We consider ligatures as separate entities just like letters and designed a set of ligature HMM’s. The two kinds of HMM’s are then combined to form a finite state network, a word model.

We have designed two script models, one for Korean Hangul characters and the other for English words [5]. The experimental recognizers based on the network models use a chain of direction codes that describes the complete locus of stylus from the first pen-down, including intermediate pen-up loci, to the last pen-up of a word. Based on this coding scheme, the networks can model both or mixed run-on and cursive words.

In the network-based approach, the character recognition is defined as the task of aligning the observation sequence optimally to the best path, which can be performed by using a dynamic programming algorithm. A great advantage of the method is that the recognizer performs letter segmentation simultaneously with recognition by fast backtracking of the best alignment.

Network-Based Approach to Online Cursive Script Recognition

Bong-Kee Sin, Jin-Yong Ha, Se-Chang Oh, and Jin H. Kim

Abstract—The idea of combining the network of HMM’s and the dynamic programming-based search is highly relevant to online handwriting recognition. The word model of HMM network can be systematically constructed by concatenating letter and ligature HMM’s while sharing common ones. Character recognition in such a network can be defined as the task of best aligning a given input sequence to the best path in the network. One distinguishing feature of the approach is that letter segmentation is obtained simultaneously with recognition but no extra-computation is required.

Index Terms—Cursive script, hidden Markov model, ligature, network search, online character recognition, segmentation, Viterbi algorithm.
The rest of the paper is organized as follows. Section II describes the basis of the approach, i.e., the use of HMM’s for letters and ligatures, and HMM networks for words. Section III addresses the recognition and segmentation algorithm. Section IV and V are dedicated to experiments using Korean Hangul and English script respectively. Finally, Section VI concludes the paper.

II. MODEL OF RECOGNIZER

The theory of statistical pattern recognition [9] states that an optimal recognition is based on the maximization of a posteriori probability \( P(W | X) \), or, equivalently by way of Bayes’ rule,

\[
P(\hat{W}, X) = \max_W P(X | W)P(W) \tag{1}
\]

where \( P(X | W) \) is the probability that a specific pattern \( X \) is observed from the word \( W \) of prior probability \( P(W) \). In order to estimate \( P(X | W) \), we have designed a network model that account for the shape and variability of handwritten word patterns. The model for \( P(W) \) is termed the language model and is separately constructed. In this study, the binary \( n \)-gram model for English words and a simple bigram model for Korean characters are used.

A. Modeling Unit

A letter model in character recognition describes the representative form of letter patterns, and is used for matching with an input. In this study we design a letter model by using the HMM. The letter HMM’s have a left-right transition structure as in Fig. 1(a) that captures the temporal characteristic of online handwriting signals. The number of states is determined based on the structure and length of input signals so that the state duration behavior can be modeled implicitly.

In unconstrained handwriting ligatures occurring at letter boundaries are the major source of shape variability. In order to model the source explicitly, let us extend the conventional concept of ligatures to include the pen-up traces (see Figs. 2 and 3) which are “invisible” but assumed to be linear. Unlike the inter-letter pen-up moves, the pen-up traces are straight like the broken lines in the figure, the shape of ligatures in general is very simple with a relatively small directional variation. Therefore a simple topology as shown in Fig. 1(b) will suffice for ligature HMM’s.

B. Word Model

A word is often described as a string of letters from an alphabet. Similarly a handwritten word can be described as an alternating sequence of letters and ligatures

\[
\text{Handwriting} = \text{Letter} \cdot \text{(Ligature} \cdot \text{Letter})^+ \tag{2}
\]

where the Letter and Ligature denote a letter and a ligature, respectively, as described in the preceding section, and “ \( \cdot \) ” and “ \( ^+ \) ” denote concatenation and repetition respectively.

Based on the above definition, a word model can be constructed as a simple concatenation of letter and ligature models. However, when one word model is defined and evaluated for each entry in the lexicon, it is impossible to design a practical real-time recognizer as the vocabulary size grows. There is a technique of structural tying that was proposed by Sin and Kim [10] where a collection of letter and ligature models are effectively combined to form a single network by sharing common models. According to their work, the structural tying depends on the word composition syntax of the letter sequence. Two types of networks are appropriate for our discussion.

The first type of network is a simple string model without any particular syntax in the letter sequence; any letter can follow any letter in the string. This type of network is the direct conversion of the definition of handwritten word. A straightforward realization is a cyclic network with two dummy nodes as shown in Fig. 4, [5] which is compared to that of connected phone or digit recognition model [11], [12]. This network allows strings of arbitrary sequence and length. Therefore a lexicon is required in order to accept only legal words. The network will be used to recognize English words in Section V.

The other type of network is the model for words involving some rigorous constraint which we call word composition syntax. There being no generic syntax across languages, however, our discussion will be made in terms of the Korean Hangul character model shown in Fig. 5. In Hangul a character represents a syllable and consists of either two or three letters. Each letter, belonging to one of three letter types, can appear only at a specific position and in a fixed order within a character. By incorporating such constraints, the resulting network will model characters better than simple string models in that no illegal character hypotheses are allowed. The Hangul character model will be explained further and tested in Section IV.

III. NETWORK SEARCH

The networks of the previous section are used to match with a given observation sequence of a script. The objective of the process is maximizing the probability of a word path and the input sequence. This is solved by the use of Viterbi algorithm, which has been shown successful in speech recognition [13], [14].

Let us denote \( w_i \) a letter or ligature HMM. Each complete path \( W = w_1 w_2 \cdots w_K \) from the start node to the end node uniquely determines a word of \( (K + 1)/2 \) letters. Given an input sequence \( X = x_1 x_2 \cdots x_T \) we can obtain the best alignment to the best path \( \hat{W} \) by applying the Viterbi algorithm that approximately maximizes the probability \( P(X | \hat{W}) \). Each alignment of \( X \) to a path \( W \) defines a segmentation

\[
S = (S(1, t_1), S(t_1 + 1, t_2), \cdots, S(t_K + 1, T))
\]

where \( S(t_i + 1, t_k) = x_{t_i+1} x_{t_i+2} \cdots x_{t_k} \) is the \( k \)-th segment corresponding to \( w_k \) and satisfies \( 0 \leq t_0 \leq t_1 \leq t_2 \leq \cdots \leq t_K = T \). Then we can rewrite the first component of the right hand side of (1) as

\[
P(X | \hat{W}) = \max_{K, W, S} P(S(t_0 + 1, t_1), \cdots, S(t_{K-1} + 1, T)[w_1 \cdots w_K]). \tag{2}
\]

Here \( w_k \)’s of odd indices are letter models, and those of even indices are ligature models.

To avoid exponential complexity in calculating (2), we assume conditional independence among the segments in \( S \). Although not valid in cursive script in general, the assumption is justified provided that each letter and ligature model describes adequately the variability of the corresponding segmental patterns. Then we can write (2) as

\[
P(X | \hat{W}) = \max_{K, W, S} \prod_{k=1}^{K} P(S(t_{k-1} + 1, t_k)|w_k). \tag{3}
\]

This equation states that the overall recognition problem can be decomposed into a sequence of subproblems of letter segmentation.
Fig. 2. Ligature patterns: (a) a pen-up ligature shown in a broken line and two small ligatures in Hangul characters and (b) three English ligatures, a long pen-up, a mixture of pen-down and pen-up, and a small pen-down.

Fig. 3. A representation of the complete pen-trajectory, including pen-down and pen-up moves. Note that the pen-up moves are distinguished by different arrows; they will be encoded using a different coding scheme.

Fig. 4. The cyclic network of HMM’s for English words.

and segment matching. Since, however, the best segmentation $S$ is not known and is highly dependent on the match result of individual segments, the maximization must be performed in terms of both the paths (determined by $K$ and $W$) and the temporal boundaries of $S$. The maximization of (3) can be called a model-wise Viterbi and written in a recursive form as

$$\Delta t(k) = \max_{s, w_k, (s \rightarrow k), t_g} \Delta t_g(g) P(S(t_g + 1, t)|w_k),$$

$$t = 1, \cdots, T$$

(4)

where $k$ denotes the $k$th node (not an HMM-state; see Fig. 6) of the current partial word path $w_1 \cdots w_g w_k$, $g$ is any node immediately preceding $k$ via the HMM $w_k$ along the link $g \rightarrow k$, and $\Delta t(k)$ is the probability of reaching the node $k$ while observing $x_1 x_2 \cdots x_1$. Note that the maximization is performed over not only the time $t_g$ (this determines the length of the last segment $S(t_g + 1, t)$) and the model $w_k$ matching to the segment, but also the preceding node $g$ which, in effect, chooses the length (in the number of nodes or unit models) of the current best partial path.

The evaluation of $P(S(t_g + 1, t)|w_k)$ in (4) comprises the recognition of a letter or a ligature given the subsequence $S(t_g + 1, t)$.

Fig. 5. Hangul character network model BongNet; null arcs shown in broken lines model the absence of the final consonant Z.

Fig. 6. Model-wise Viterbi computation in an FSN.
It is performed by the Viterbi algorithm using an HMM in parallel with the above model-wise Viterbi as

$$
\delta_t^m(j) = \max_{1 \leq i \leq d} \delta_{t-1}^m(i) a_{ji}^m b_{ji}^m(x_t),
$$

where the superscript \(m\) identifies the HMM under consideration, \(a_{ji}^m\) and \(b_{ji}^m()\) are the state transition and the output parameters respectively of the model, and \(\delta_t^m(j)\) is the joint probability of aligning the partial symbol sequence thus far observed and the best partial state sequence reaching the state \(j\) (see Fig. 7).

In Fig. 8 the alignment information is kept in \(\psi_t^m(j)\), the pointer to the preceding state at time \(t-1\) for each state \(j\) of model \(m\) at time \(t\). \(\phi_t^m(j)\) keeps the time passed inside the model \(a_{ji}^m\), which corresponds to the length of the partial segment.

The computation done at the nodes linking HMM's is similar to the maximization done at HMM-internal states. For each time \(t\) and node \(k\), the maximization is performed over all models \(m\) of all arcs \(g \rightarrow k\) as

$$
\Delta_t(k) = \max_{m = 1}^{M} \delta_t^m(N_m)
$$

where \(\delta_t^m(N_m)\) is the likelihood of the final state \(N_m\) of the model \(m\) at time \(t\). The alignment information is maintained in the pair of \(\Psi_t(k)\) and \(\Omega_t(k)\), each storing the pointer to the best preceding node \(\tilde{y}\) and the pointer to the most likely model \(\tilde{m}\) between nodes \(\tilde{y}\) and \(k\). \(\Psi_t(k)\) is the length of the subsequence \(x_{t-\phi_t(k)-1} \cdots x_{t}\) which is aligned to the state sequence of the model \(\tilde{m}\). When the forward pass is over, we can then obtain the optimal word label (as the recognition result) and the best segmentation by backtracking the optimal path which is facilitated by quantum jumps of \(\Phi_t(k)\)-long steps.

The majority of the computation time is spent in the loop of forward network search of Fig. 8. Let \(L\) be the number of HMM's in a network. In addition let \(N\) be the average number of states in an HMM, and \(l\) be the average number of incoming transitions per state. As we are using left-right HMM's, the computation time for an HMM is \(\frac{LNT}{M_{NT}}\) instead of \(\frac{LNT}{NT}\) for an ergodic model. The forward pass is done for every HMM in the network. Therefore, the total computation amounts to \(LM_{NT}\), which is linear in \(T\) and favors \(M\) that is as small as possible.
TABLE I
TRAINING DATA AND TEST DATA SETS

<table>
<thead>
<tr>
<th>Data</th>
<th>Data sets</th>
<th>Writers, samples in total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td>H</td>
<td>37 writers, 45,000 chars</td>
</tr>
<tr>
<td>Data</td>
<td>K</td>
<td>21 writers, 7,000 chars</td>
</tr>
<tr>
<td>Test</td>
<td>H'-dep</td>
<td>11 adapted writers, 4,900 chars</td>
</tr>
<tr>
<td>Data</td>
<td>K'-dep</td>
<td>21 adapted writers, 1,900 chars</td>
</tr>
<tr>
<td></td>
<td>H'-ind</td>
<td>10 new writers, 8,100 chars</td>
</tr>
<tr>
<td></td>
<td>K'-ind</td>
<td>8 new writers, 1,700 chars</td>
</tr>
</tbody>
</table>

TABLE II
PERFORMANCE OF CHARACTER MODELS IN % CORRECT RECOGNITION

<table>
<thead>
<tr>
<th>Model(train set)</th>
<th>H'-dep</th>
<th>H'-ind</th>
<th>K'-dep</th>
<th>K'-ind</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Lambda_8(H) )</td>
<td>84.76</td>
<td>84.24</td>
<td>82.83</td>
<td>84.99</td>
</tr>
<tr>
<td>( \Lambda_{16}(H) )</td>
<td>86.52</td>
<td>87.82</td>
<td>81.30</td>
<td>83.27</td>
</tr>
<tr>
<td>( \Lambda_8(H+K) )</td>
<td>84.95</td>
<td>86.12</td>
<td>84.56</td>
<td>88.55</td>
</tr>
<tr>
<td>( \Lambda_{16}(H+K) )</td>
<td>88.03</td>
<td>88.10</td>
<td>86.24</td>
<td>87.43</td>
</tr>
</tbody>
</table>

inward arcs and the outward arcs respectively. The mesh-like arcs between the two columns represent the ligature classes. The network has in total 40 ligature models, half \( L_{\text{cap}} \)'s and half \( L_{\text{cap}} \)'s.

B. Experimental Context

For model training and testing, we have used two data sets \( H \) and \( K \), as specified in Table I. The writers of \( H \) are high school students, while those of \( K \) are college students writing more cursively.

To train letter and ligature models, we have manually segmented the character samples of the training sets into letters and ligatures. Each of the segments is resampled in space including the pen-up movements, and converted to a chain of direction codes. HMM’s are trained by the forward/backward algorithm [15] using the collection of the chain codes.

Test sets \( H' \)'s and \( K' \)'s are separately prepared by adapted and nonadapted subjects as in Table I where adaptation means that the test set writers have contributed to the training set.

C. Character Recognition Result

The first set of tests concern the performance of two coding schemes of eight and sixteen direction codes. For each encoding we have created two models with two training sets \( H \) and \( H + K \), in total four models. The recognition results with the single best candidates on test sets \( H' \)'s and \( K' \)'s are summarized in Table II. Recognition results on the test sets \( H' \)-dep and \( H' \)-ind clearly show the advantage of using sixteen direction codes as is our expectation based on the quantization error in data coding.

In the case of \( K' \) sets, however, \( \Lambda_{16}(H) \)'s performance is lower than that of \( \Lambda_8(H) \)'s. Whereas the results with the models \( \Lambda_{16}(H + K) \) and \( \Lambda_8(H + K) \) shows mixed results: \( \Lambda_{16}(H + K) \) is superior on \( K' \)-dep by +1.68% correct recognition, but still inferior on \( K' \)-ind by −1.12% although the gap has been narrowed from −1.72%. We attribute the latter result partially to undertraining of HMM’s or, more precisely, to the data characteristic as remarked previously, i.e., the general stylistic difference between the subject groups.

The other set of tests involves smoothing the output parameters of the HMM’s to alleviate the problem of overfitting to the given training set [16]. According to the test results with the smoothed models as shown in the Table III, performance improvement of \( \Lambda_{30} \)'s is highly conspicuous, a strong evidence to undertraining and a weak evidence to the bias of the training set.

D. Decoding Result

One distinctive feature of the proposed approach is that character (or word) recognition and letter segmentation is obtained simultaneously as a result of network searching. The recognition result is the character label that is defined by the letter sequence uncovered by backtracking triggered at the final node. A set of optimal letter boundaries are obtained from the same result of backtracking.

Let \( \text{parus} \) consider an input sequence \( x_1x_2\cdots x_{56} \) [see Fig. 9(a)] for a Hangul character and the corresponding network path that has been computed the most likely and is highlighted in bold arrows and circles [Fig. 9(b)]. By backtracking the path we can obtain a desired symbol-to-state alignment as shown in Fig. 9(c).
Fig. 10 shows several examples of segmentation results mapped to the raw handwriting. Letter segmentation is easy if it is known that the characters are discrete and regular as in Fig. 10(a). But the points lying on the smooth cursive strokes as in Fig. 10(b) are next to impossible to detect without using letter-specific and character-specific knowledge. In the proposed model, however, it is straightforward and those points are determined in simple probabilistic terms. Let us now examine some examples in Fig. 11 which summarizes the failure modes. In Fig. 11(a) the characters are segmented correctly but classified incorrectly. In Fig. 11(b) and (c) the recognition failed in both ways, but the latter results are reasonable to human readers because they are highly ambiguous.

V. ENGLISH WORD EXAMPLE

A. Network of Interconnected HMM’s

The English word model as shown in Fig. 4 has a circular structure with backward connection of ligature models, implementing the repetition of letters as in the definition of handwritten word in Section II. The current network involves 57 HMM’s in total.
B. Word Recognition

Since no word composition syntax has been incorporated in the word model, the DP algorithm described in Section III allows a large number illegal words. So the forward pass of the algorithm has been modified to yield several word paths from which any illegal word paths are pruned out at each step of local hypothesis generation by consulting a lexicon in parallel.

Prior to starting the recognition procedure, every input stroke is labeled as BODY or ADDENDUM according to their size and shape based on heuristics. BODY includes the major strokes comprising a word as shown in Fig. 12. ADDENDUM includes the dot in “i,” “j,” “t,” and “x.” They were trained using samples with delayed strokes removed. During recognition, the models also participate in the generation of letter hypotheses.

When the forward Viterbi pass is over, several candidate word paths are determined and their letter segmentations are also obtained. By referring to the horizontal position of each ADDENDUM, the delayed strokes are added to the appropriate “incomplete” letter segments as in Fig. 14. Then the corresponding complete HMM’s are reevaluated using the complete segments. Finally, the word hypotheses are reordered according to the new probabilities. The overall recognition procedure is given in Fig. 15.

C. Experiments

For training HMM’s, we have prepared isolated samples of 2194 print and 2045 cursive letters respectively, and hand-segmented 6763 letters and 4505 ligatures from unconstrained word samples of eight subjects.

The test set used for letter recognition consists of 1550 printed letters and 1548 cursive letters written in boxes, and 2048 letter segments from word samples. They were collected from three subjects independent of the training set. The test result summarized in Table IV shows 88.74% correct recognition with the single best choice and 98.59% with top five candidates.

In word recognition, a lexicon of about 25,000 entries were employed. Table V shows word recognition rates with and without ligature modeling. For total 728 and 71 distinct words written by three writers, we obtained 80.49% correct recognition with the single best candidate and 90.66% with top five candidates. Ligature modeling improved the recognition rate by 6.31%, reducing the error rate by 24.44%. When the ligature model was removed and the ligature pattern was included as a part of letters, the recognition rate dropped to 74.18% from 80.49%. The difference implies the utility of the ligature modeling.

Fig. 16 illustrates letter/ligature segmentation examples. We consider that the segmentation results are particularly stimulating because the network structure is very simple and there is only one ligature HMM.

VI. CONCLUSION

The network-based modeling technique for time sequential online word patterns has been described with an explicit commitment to extended ligatures. The technique can be applied to any alphabet-based languages or even multiple languages, and possibly to word sequence recognition with a few extensions.

Experimental results using English and Korean scripts have confirmed us that the letter segmentation problem can be solved with
elegance. There is no extra computation required. The recognition algorithm is a simple DP except for the treatment of delayed strokes. To summarize, the proposed approach constitutes a consistent framework, both in knowledge representation and computation algorithm.

REFERENCES


