Proc. ICDAR, to appear, Aug. 2003

Handwriting Recognition Using Position Sensitive Letter N-Gram Matching

Adnan El-Nasan, Sriharsha Veeramachaneni, George Nagy
DocLab, Rensselaer Polytechnic Institute, Troy, NY 12180
elnasan@rpi.edu

Abstract

We propose further improvement of a handwriting recognition method that avoids segmentation while able to recognize words that were never seen before in handwritten form. This method is based on the fact that few pairs of English words share exactly the same set of letter bigrams and even fewer share longer n-grams. The lexical n-gram matches between every word in a lexicon and a set of reference words can be precomputed. A position-based match function then detects the matches between the handwritten signal of a query word and each reference word. We show that with a reasonable set of reference words, the recognition of lexicon words exceeds 90%.

1. Introduction

We are proposing a single-user unconstrained handwriting recognition system that utilizes partial word matching to detect letter-bigram or longer segments from a feature-based representation of word patterns. The system has a *lexicon*, and a *reference set*. The lexicon is the set of all plausible word labels. Words in the reference set are words from the lexicon for which we have handwritten samples. The proposed system consists of three stages: lexical processing, signal matching and classification. The *lexical processing* stage pre-computes the bigram match properties for each word in the lexicon by matching the label of a lexicon word against the label of each reference word. The *signal matching* stage reports the length of the longest matching segment between the feature representation of the unknown and the feature representation of each reference word. In contradistinction to our earlier work [4], these matches are limited to the positions where each lexical candidate matches the label of a reference word. The *classification* stage then finds a label from the lexicon that has lexical match properties that most resemble the signal match properties of the unknown. This method is similar in principle to the error correcting output codes proposed by Dietterich and Bakiri for solving multi-class learning problems [1]. In our method, each reference word induces a dichotomy on the lexicon and therefore the error correcting property is based on similarity between segments of lexicon and reference words.

A letter n-gram is a sequence of *n* consecutive letters. N-grams have been studied and utilized since the sixties. Raviv introduced Markov models to OCR [9] and Shinghal and Toussaint applied the Viterbi algorithm [10][11]. Hull and Srihari quantized n-grams probabilities [7] and combined them with dictionary lookup [8]. Suen tabulated the growth in the number of distinct n-grams as a function of vocabulary size [12]. The entropy of n-grams for $n \le 5$ is computed in [14].

Hong and Hull introduced partial-word matching and used them for detecting patterns from the same source with similar shapes [5][6]. Feature-based partial-word matching for detecting bigram co-occurrences combines some of the advantages of character-based and word-based recognition. Like character-based recognition, vocabulary is expandable and recognition is not limited to words with explicit handwritten samples in a training set. However, feature-based bigram detection is more stable than character-based recognition because it avoids segmentation and special ligature processing.

We introduced letter n-gram co-occurrences between pairs of words for word discrimination in [2]. We have shown in [3] that with a reasonable number of reference words, bigrams represent the best compromise between the recall ability of single letters and the precision of trigrams. We also determined the performance of an ideal system as a function of lexicon and reference set sizes. In [4] we proposed a complete handwriting recognition system based on detecting bigram co-occurrences and reported its performance as a function of lexicon and reference set sizes.

Like the widely used Hidden Markov methods, feature-level bigram detection brings context into the recognition stage instead of relegating it to post-processing. Unlike HMM, it requires the estimation of relatively few parameters, storing instead a reference set of representative patterns and their labels.

2. Method and notation

The proposed system is based on detecting *match properties* between words (Figure 1). The match properties indicate the presence or absence of at least a bigram-length match between the lexical labels of two words.

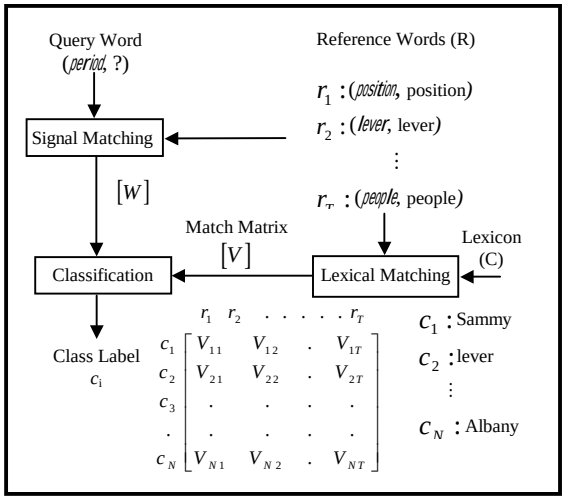

Figure 1 Data flow

The system uses a lexicon and a reference set. The match properties of each lexicon word, for a specific reference set, are pre-computed at the lexical stage. The expected signal match properties of an unknown word are computed at the signal stage using length and location information about n-gram matches computed at the lexical stage. This information improves word discrimination by eliminating the possiblity of false matches at the "wrong" location. At the classification stage, the label of the lexical word with the match properties that are most similar to the match properties of the query word is assigned as the label of the query. To describe the system in detail, we are using the following notation:

C : the lexicon of length N.
c_i : the ith lexicon word.
R : the reference set of length T.
r_j : the jth reference word.
v_{ij}^k : the length and position of the kth lexical match (of at least bigram length) between c_i and r_j $v_{ij}^k = \left(l_{ij}^k, s_i^k, s_j^k\right)$.
l_{ij}^k : the number of letters in the kth match.
$\hat{l}_{ij}^k$: the estimated length of the kth ink match.
s_i^k : Index of letter in c_i where match begins.
s_j^k : Index of letter in r_j where match begins.
V_{ij} : $\left\{v_{ij}^k : \forall k\right\}$.
V : $\left[V_{ij}\right]$.
w_{ij}^k : the estimated length and position of the kth ink match between the query word q, hypothesized as c_i, and r_j, $w_{ij}^k = \left(\hat{l}_{ij}^k, \hat{s}_i^k, \hat{s}_j^k\right)$.
W_{ij} : $\left\{w_{ij}^k : \forall k\right\}$.
W : $\left[W_{ij}\right]$.
$R(c_i)$: $\left\{W_{xy} : x = i, 1 \le y \le T\right\}$.
R^U : $\left\{W_{xy} : 1 \le x \le N, 1 \le y \le T\right\}$.
$\bar{R}(c_i)$: $R^U - R(c_i)$.

2.1. Lexical processing

Given a lexicon C and a reference set R, the match properties matrix V is calculated. Each element V_{ij} corresponds to all (bigram or longer) lexical matches between the lexical candidate c_i and the reference word r_j. Each of these matches v_{ij}^k describes the length of the match l_{ij}^k and its shift position s_i^k and s_j^k, in both c_i and r_j. These vectors are defined only when a lexical match exists between lexicon and reference words. An example V is shown in Table 1.

Table 1 Example of a match matrix

Reference / Lexicon	adds	lever	beeper
adds	4,1,1	0	0
beer	0	2,3,4	3,1,1; 2,3,5
leopard	0	2,1,1	0
leopards	2,7,3	2,1,1	0
lever	0	5,1,1	2,4,5
mere	0	2,2,4	2,2,5
beeper	0	2,5,4	6,1,1

2.2. Signal processing

In this stage the match properties of the unknown word q are detected. This is done by a location-guided matching of the feature representation of q and the feature representation of each handwritten word in the reference set.

Each handwritten word is represented as a string of feature symbols. These features are very simple and represent extremal points, cusps and intersections of the trace of the stylus in the x and y directions. Flybacks are detected and intersection points with the original stroke are marked. Each of these features is assigned an alphabetic label (Figure 2).

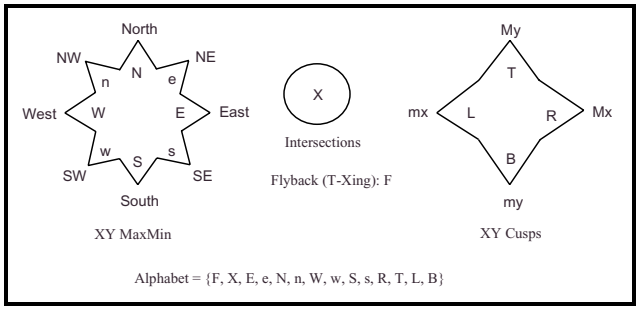

Figure 2 Features and their labels

The string representation of the word is constructed by analyzing its coordinate sequence and concatenating the corresponding feature labels (Figure 3).

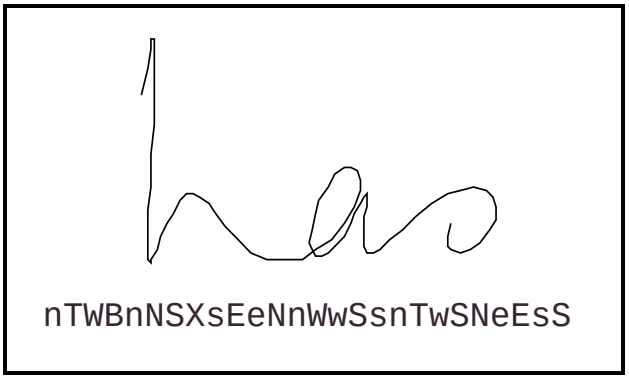

Figure 3 The feature string of *has*

2.2.1. Letter feature-length estimation. The expected feature-length of the letters in the alphabet is calculated by modeling the reference words and their feature lengths as an over-determined system of linear equations.

The total length of each word is the sum of its constituent character lengths. A linear equation is constructed for every word in the reference set and a least-squares solution is found for the whole system (Figure 4). A more detailed description is found in [13].

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 2 & \cdots & 0 \\ 1 & 1 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 0 & 0 & 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} L_a \\ L_b \\ \vdots \\ L_z \\ L_A \\ \vdots \\ L_Z \end{bmatrix} = \begin{bmatrix} L_{have} \\ L_{lever} \\ L_{bare} \\ \vdots \\ \vdots \\ \vdots \\ L_{Zebra} \end{bmatrix}$$

Figure 3 A system of linear equations to estimate feature-length of the alphabet

2.2.2. Detecting ink matches. The longest common subsequence (LCS) between the unknown and each reference word, near the expected location, is now determined. We align the reference word's label and the query's hypothesized label with their feature string representation from the left and right ends. We localize the search for the LCS to that part of string alignment cost matrix that corresponds to the rectangular intersection of the alignments. Figure 5 shows that the *r*'s are the longest match, so *st* is not detected as the desired matching segment between the words *history* and *request*. Such false matches are avoided by localizing the search in the cost matrix to the estimated location of the bigram *st*.

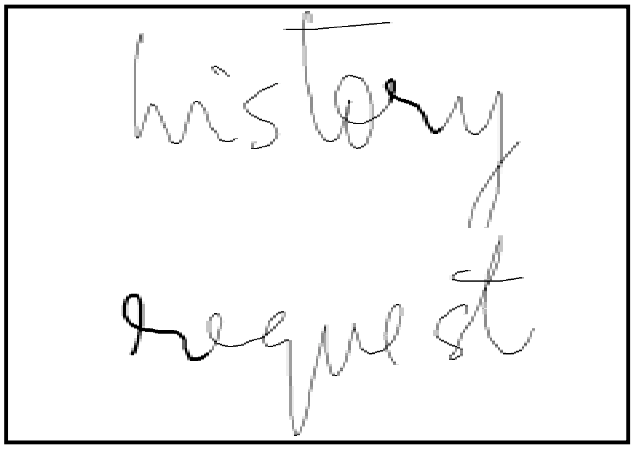

Figure 4 Matching segments between the words *history* and *request*

Detecting common bigrams or longer segments between handwritten words is inherently ambiguous. Therefore, we model the process as a probabilistic two-class detection problem: *matches* (M) and *no matches* $(\bar{M})$.

2.3. Classification

We can formulate the classification problem of choosing the lexical word c_i, represented with respect to the reference set by V, given the query's match matrix W, as:

$$\begin{aligned} P(c_i|W) &= \frac{P(W|c_i)P(c_i)}{P(W)} \\ &= \frac{P(c_i)}{P(W)} \prod_{R^U} P\left(w_{ij}^k|c_i\right) \\ &\propto \prod_{R^U} P\left(w_{ij}^k|c_i\right) \\ &\equiv \prod_{R(c_i)} P\left(l_{ij}^k \middle| \hat{l}_{ij}^k, M\right) \prod_{\bar{R}(c_i)} P\left(l_{ij}^k \middle| \hat{l}_{ij}^k, \bar{M}\right) \end{aligned}$$

$P\left(w_{ij}^k|c_i\right)$ is the probability that query word q and reference word r_j exhibit match properties represented by w_{ij}^k, where q has the same lexical label as c_i. Therefore, $P\left(w_{ij}^k|c_i\right) = P\left(l_{ij}^k \middle| \hat{l}_{ij}^k, M\right)$ if c_i has a lexical match with r_j, and $P\left(w_{ij}^k|c_i\right) = P\left(l_{ij}^k \middle| \hat{l}_{ij}^k, \bar{M}\right)$ otherwise.

$P\left(l_{ij}^k \middle| \hat{l}_{ij}^k, M\right)$ is the probability that, given a lexical match exists between c_i and r_j, query word q's *k*th match with reference word r_j has a length l_{ij}^k where the expected length of the match between c_i and r_j is $\hat{l}_{ij}^k$.

The query word q represented by its match matrix W will be classified to class c^*, where:

$$c^* = \arg\max_i \prod_{R(c_i)} P\left(l_{ij}^k \middle| \hat{l}_{ij}^k, M\right) \prod_{\bar{R}(c_i)} P\left(l_{ij}^k \middle| \hat{l}_{ij}^k, \bar{M}\right)$$

$$c^* = \arg\max_i \frac{1}{\prod_{R^U} P\left(l_{ij}^k \middle| \bar{M}\right)} \prod_{R(c_i)} P\left(l_{ij}^k \middle| \hat{l}_{ij}^k, M\right) \prod_{\bar{R}(c_i)} P\left(l_{ij}^k \middle| \hat{l}_{ij}^k, \bar{M}\right)$$

$$c^* = \arg\max_i \frac{1}{\prod_{R(c_i)} P\left(l_{ij}^k \middle| \bar{M}\right) \prod_{\bar{R}(c_i)} P\left(l_{ij}^k \middle| \bar{M}\right)} \prod_{R(c_i)} P\left(l_{ij}^k \middle| \hat{l}_{ij}^k, M\right) \prod_{\bar{R}(c_i)} P\left(l_{ij}^k \middle| \hat{l}_{ij}^k, \bar{M}\right)$$

$$c^* = \arg\max_i \prod_{R(c_i)} \frac{P\left(l_{ij}^k \middle| \hat{l}_{ij}^k, M\right)}{P\left(l_{ij}^k \middle| \hat{l}_{ij}^k, \bar{M}\right)}$$

The class-conditional distributions $P\left(l_{ij}^k \middle| \hat{l}_{ij}^k, M\right)$ and $P\left(l_{ij}^k \middle| \hat{l}_{ij}^k, \bar{M}\right)$ are modeled as discrete 2-D empirical distributions of the feature-based match lengths among the words of the reference set. The probability of match location is conditioned on $\hat{l}_{ij}^k$, the shorter edge of the rectangle enclosing plausible matches in the string alignment cost matrix.

3. Experiments and results

A database of about 6000 words was written by a single writer, with no constraints, on a CrossPad. We selected eleven mutually exclusive sets of samples (words ranging from 5 to 15 characters): a reference set *RSet*, and 10 test sets *TSet-i*. Less than 50% of distinct word labels appear in both *RSet* and any *TSet-i*. Table 2 describes the statistics of these datasets. The last column of Table 2 is an average over the ten test sets.

Table 2 Statistics of data used in testing

	Database	RSet	TSet-i
Number of words	5940	1000	100
Lexically unique	1661	674	66
Characters/word (average)	1-25 (4.3)	5-15 (7.32)	5-15 (7.33)

3.1 Preliminary results

We study the effect, on the system performance, of adding new words to the reference set and to the lexicon. Each word in the ten test sets will be used as a query word. A match vector will be generated for each query.

Table 3 reports the recognition results for each test set as a function of three different sizes of reference set and lexicon. Figure 6 shows how the average accuracy, over all test sets, increases as the number of reference words increases. Figure 7 shows how the average accuracy decreases as the size of the lexicon increases, given a fixed number of reference words.

Table 3 Recognition rates as a function of reference set and lexicon sizes

	Reference =100			Reference= 500			Reference=1000		
	Lex 100	Lex 500	Lex 1000	Lex 100	Lex 500	Lex 1000	Lex 100	Lex 500	Lex 1000
TSet-1	70	52	43	77	70	61	84	72	64
TSet-2	81	64	56	88	73	70	90	75	71
TSet-3	75	59	49	89	72	62	92	78	69
TSet-4	74	63	57	90	86	79	93	87	84
TSet-5	73	57	54	88	71	66	85	74	68
TSet-6	76	61	52	79	67	59	85	76	72
TSet-7	77	58	50	92	80	74	95	84	77
TSet-8	81	66	60	88	79	73	92	82	79
TSet-9	70	52	48	88	77	71	90	78	70
TSet-10	70	54	46	84	66	59	85	71	67

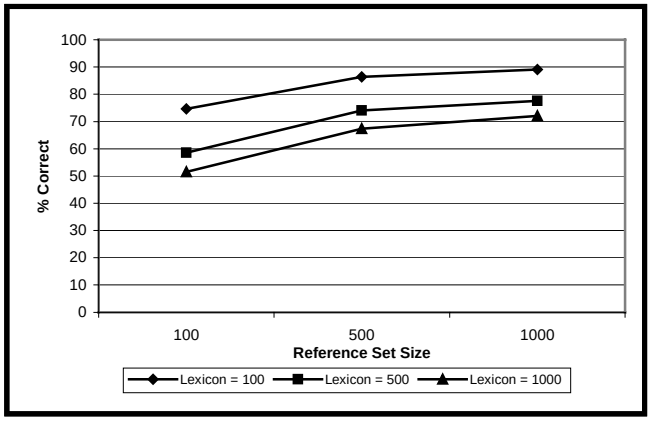

Figure 5 Average accuracy as a function of the number of reference words

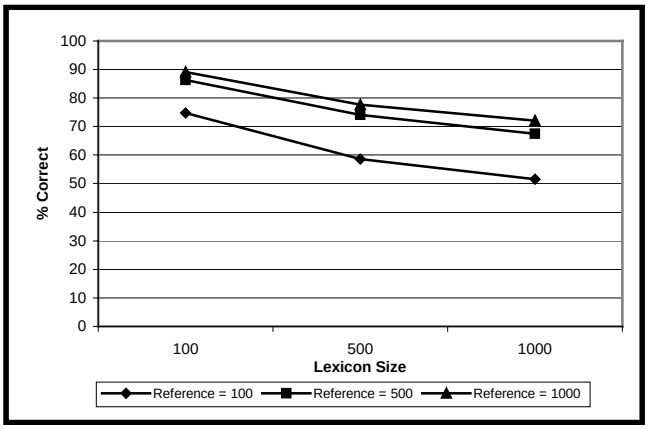

Figure 6 Average accuracy as a function of the lexicon size

4. Discussion

The accuracy of the system improves as the number of reference words increases because additional reference

words compensate for matching errors due to letter-form or stroke variations. As the size of the lexicon increases, given a fixed reference set, the accuracy decreases as a result of attempting to pack more samples into the fixed-size feature space. The results are substantially as predicted from simulations [3].

Bigram detection using information about match position and expected length improves significantly on the accuracy we reported in [4]. We are still using an elementary set of features and simplistic string matching. We are currently modifying the features and signal matching routines to improve the estimation of the class-conditional distributions. We plan to use features that are more expressive, and implement more elaborate approximate string matching.

Table 3 shows increase in recognition rates, for all test sets and lexicon sizes, as the number of reference words increases. We believe that some sets improve more than others because they contain words with higher average Hamming Distance. Therefore, the words tolerate more match errors incurred at the signal matching stage. We are currently studying the relation between the lexical and signal matching stages and their contribution to the overall accuracy of the recognition system. The reference set can be easily augmented by adding newly recognized words. This provides a practical means for improving accuracy through adaptation.

At the signal matching stage, we assume independence between matches of the unknown and each reference word. When two reference words share the same letters with the unknown then these matches are correlated, which biases the classifier in favour of lexicon words that contain frequent bigrams. We intend to model such correlations to improve accuracy.

We will make use of standard word frequencies to resolve multiple candidates. These will eventually be modified to account for the writer's own word-usage statistics. We will consider using dynamic word-transition models as well.

5. Acknowledgment

We thank Yarmouk University, Jordan, for their financial support. We are grateful to IBM's Pen Computing Group for providing the data and also for their valuable comments and discussions. We thank Professors Frank LeBourgeois and Angelo Marcelli for their suggestions and help on feature extraction.

6. References

[1] T. G. Dietterich, G. Bakiri, "Solving multiclass learning problems via error-correcting output codes," *Journal of Artificial Intelligence Research*, vol. 2, pp. 263-286, 1995.

[2] A. El-Nasan, G. Nagy, "Ink-Link," *Proceedings of the 15th International Conference on Pattern Recognition*, vol. 2, pp. 573-575, Barcelona, 2000.

[3] A. El-Nasan, S. Veeramachaneni, G. Nagy, "Word discrimination based on bigram co-occurrences," *Proceedings of the 6th International Conference on Document Analysis and Recognition*, pp. 149-153, Seattle, 2001.

[4] A. El-Nasan, G. Nagy, "On-Line handwriting recognition based on bigram co-occurrence," *Proceedings of the 16th International Conference on Pattern Recognition*, vol. 3, pp. 740-743, 2002.

[5] T. Hong, J. Hull, "Character segmentation using visual inter-word constraints in a text page," *Proceedings of the SPIE - The International Society for Optical Engineerin*g, vol. 2422, pp.15-25, 1995.

[6] T. Hong, J. Hull, "Visual inter-word relations and their use in OCR post-processing," *Proceedings of the Third International Conference on Document Analysis and Recognitio*n, pp. 442-445, 1995.

[7] J. Hull, S. Srihari, "Experiments in text recognition with binary n-grams and Viterbi algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligenc*e, vol. 4, no. 5, pp. 520–530, 1982.

[8] J. Hull, S. Srihari, R. Choudhari, "An integrated algorithm for text recognition: comparison with a cascade algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligenc*e, vol. 5, no. 4, pp. 384–395, 1983.

[9] J. Raviv, "Decision making in Markov chains applied to the problem of pattern recognition," *IEEE Transactions on Information Theory*, vol. 3, no. 4, pp. 536-551, 1967.

[10] R. Shinghal, G.T. Toussaint, "Experiments in text recognition with the modified Viterbi algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 1, no. 2, pp. 184–193, 1979.

[11] R. Shinghal, G.T. Toussaint, "The sensitivity of the modified Viterbi algorithm to the source statistics," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 2, no. 2, pp. 1181–1184, 1980.

[12] C.Y. Suen, "N-gram statistics for natural language understanding and text processing," *IEEE Transactions on Pattern Analysis and Machine Intelligenc*e, vol. 1, no. 2, pp. 164–172, 1979.

[13] Y. Xu, G. Nagy, "Prototype extraction and adaptive OCR," *IEEE Transactions on Pattern Analysis and Machine Intelligenc*e, vol. 21, no. 12, pp. 1280–1296, 1999.

[14] E. Yannakoudakis, G. Angelidakis, "An insight into the entropy and redundancy of the English dictionary," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 6, pp. 960-970, 1988.