On the Dependence of Handwritten Word Recognizers on Lexicons

Hanhong Xue, *Student Member*, *IEEE*, and Venu Govindaraju, *Senior Member*, *IEEE*

Abstract—The performance of any word recognizer depends on the lexicon presented. Usually, large lexicons or lexicons containing similar entries pose difficulty for recognizers. However, the literature lacks any quantitative methodology of capturing the precise dependence between word recognizers and lexicons. This paper presents a performance model that views word recognition as a function of character recognition and statistically "discovers" the relation between a word recognizer and the lexicon. It uses model parameters that capture a recognizer's ability of distinguishing characters (of the alphabet) and its sensitivity to lexicon size. These parameters are determined by a multiple regression model which is derived from the performance model. Such a model is very useful in comparing word recognizers by predicting their performance based on the lexicon presented. We demonstrate the performance model with extensive experiments on five different word recognizers, thousands of images, and tens of lexicons. The results show that the model is a good fit not only on the training data but also in predicting the recognizers' performance on testing data.

Index Terms—Handwriting recognition, word recognition, performance prediction, performance model, multiple regression.

1 Introduction

The field of offline handwritten word recognition has advanced greatly in the past decade. Many different approaches have been proposed and implemented by researchers [1], [2], [3], [4], [5], [6]. In the literature, performance of the handwritten word recognizers is generally reported as accuracy rates on lexicons of different sizes, e.g. 10, 100, and 1,000. We believe this characterization is inadequate because, besides the lexicon size, the performance depends on other factors as well, such as the nature of the recognizer and the quality of the input image.

It is commonly expected that word recognition with larger lexicons is more difficult [1], [2], [3], [4], [5], [6]. Marti and Bunke [7] report the influence of vocabulary size and language models on handwritten text recognition by using a wide range of lexicon sizes and several language models. Their results confirm that larger vocabularies are more difficult when language models are involved. However, lexicon size can be an unreliable predictor because it ignores the similarity between lexicon words. A lexicon containing 10 similar words is much more difficult than another containing 10 completely different words (from the viewpoint of the word recognizer). Therefore, besides lexicon size, a performance model must also consider the similarity between lexicon entries.

String edit distance, defined as the minimum number of insertion, deletion, and substitution operations required to convert one string to another, is often used as a similarity measure for strings. However, it depends only on the strings and does not take into account the nature of the recognizer or the writing style of script. In order to make the edit distance suitable for handwriting applications, researchers have used the generalized edit distance based on units that are more granular than characters, such as strokes or graphemes, and additional edit operations, such as splitting, merging, and group substitution [8], [9], [10]. Generalized edit distances improve accuracy by measuring similarity between words in more details but costs additional processing time.

Another possible measure of recognition difficulty is perplexity which is widely used in evaluating language models [11], [12], [7]. After all, the lexicon can be considered as a language model which enumerates all of the strings it accepts. (Use of other models such as N-Gram, only results in supersets of the lexicon and not exactly the lexicon.) Generally speaking, perplexity is the average number of possible *successors* of any sequence of observations. When applied to a sequence of characters, it considers words sharing prefixes but ignores words sharing suffixes. For example, two lexicons $\{\mathrm{as}, \mathrm{of}\}$ and $\{\mathrm{as}, \mathrm{os}\}$ will result in the same perplexity when all entries have the same a priori probability, but to most word recognizers the first lexicon is easier than the second one. Thus, perplexity is not adequate for the purpose of measuring recognition difficulty by a lexicon.

Grandidier et al. [13] have studied the influence of word length on handwriting recognition. They conclude that it is easier to recognize long words than short words and lexicons consisting of long words are less difficult than those consisting of short words. In their experiments, both recognition rate and relative perplexity, which is based on a posteriori probabilities output by a recognizer, are used to measure the difficulty of the recognition task. It should be noted that both recognition rate and relative perplexity are not available before recognition is performed, hence, rendering them useless in *predicting* accuracy.

Image quality is critical to image pattern recognition tasks including word recognition. The first subtask is to find quantitative measures of image quality. One possibility is the use of parameterized image defect models [14], [15], where

• *The authors are with the Center of Excellence for Document Analysis and Recognition (CEDAR), Department of Computer Science and Engineering, State University of New York at Buffalo, Buffalo, NY 14260. E-mail: {hxue, govind}@cedar.buffalo.edu.*

Manuscript received 19 Nov. 2001; revised 25 Mar. 2002; accepted 13 June 2002.
Recommended for acceptance by M. Pietikainen.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 115413.

image size, resolution, skew, blur, binarization threshold, pixel sensitivity, and other parameters are used to characterize image quality and to generate pseudoimages. The defect models have been applied to the evaluation of OCR accuracy on synthetic data [16], [17]. However, to the best of our knowledge, there has been no application reported on evaluation of handwritten word recognizers.

The common theme of most of the previous work on the topic has been to base the prediction of performance on experimental results. This approach allows us to only observe the tendency of performance change when performance parameters are altered because no quantitative modeling directly associates performance with parameters. Thus, the models based purely on empirical results leave questions such as *"Is the relationship quadratic, exponential, etc.?,"* unanswered.

In an attempt to more accurately measure the difficulty of recognition tasks, lexicon density, a measure that combines the effect of both the lexicon size and the similarity between words, has been previously presented by the authors [10]. A new generalized edit distance, namely *slice distance*, is calculated on two word models that use character segments. Then, lexicon density is defined as the product of two quantities: 1) the reciprocal of the average slice distance obtained on the given lexicon, and 2) an empirically chosen function of lexicon size. Experimental results have shown an approximate linear relation between lexicon density and recognition accuracy. Continuing upon this work, we have proposed using multiple regression models instead of choosing performance function empirically [18].

Our previous work focuses on the calculation of the distance between two word models based on the inner representation of a word recognizer and does not come up with a rigorous performance model to associate model distance with recognition accuracy. Besides the lack of a performance model, another disadvantage is the complexity of calculating model distances. Since different recognizers have different definitions of word models, model distance depends necessarily on the recognizer and can be as complex as the recognition mechanism itself, thus, making it unsuitable in real-time applications.

To overcome these disadvantages, this paper proposes a performance model that can be generalized to all word recognizers that are based on character recognition. Leaving out the details of recognizer-dependent word models, we calculate the simple string edit distance [19] of two words in their alphabetic forms which are considered as the ultimate abstractions of word models. Then, the edit distance between a nontruth word and the truth can be viewed as the evidence of not choosing the nontruth. When the recognizer totally ignores this evidence, a misclassification occurs. Based on this idea, this paper mathematically derives a performance model and converts it into a multiply regression model (Section 2). In Section 3, experiments are described on five different word recognizers running on 3,000 postal word images with tens of lexicons, not only to decide model parameters but also to verify the accuracy of the model. Section 4 presents the analysis of recognizers in terms of model parameters, the interpretation of influence of word length, and the possible use of distance measures other than edit distance in the performance model. Section 5 presents conclusions and future research directions.

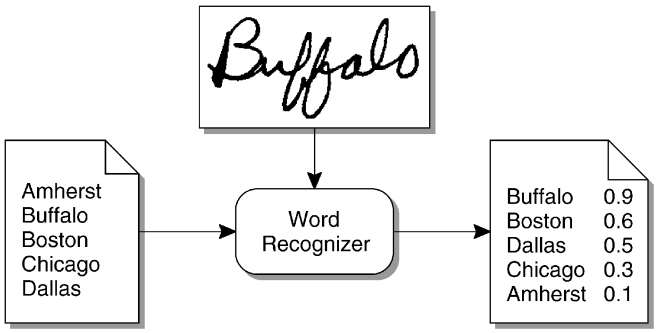


Fig. 1. Lexicon-driven word recognizer as a black box.

2 The Performance Model

Our objective is to build a quantitative model to associate word recognition performance with lexicons to allow the prediction of performance. Once the form of the model is derived, regression analysis can be applied to determine the model parameters. The form of the model must certainly depend on the performance factors it accommodates. However, it is difficult to consider exhaustively, all of the different factors simply because they are too many. Therefore, before deriving the model, we need to examine which of the factors should be considered and how they affect the word recognizer performance.

2.1 Performance Factors

The task is to derive a model with the ability to predict performance for any word recognizer. Thus, the model must be able to treat the recognizer as a black box. Fig. 1 illustrates the black box word recognizer.

Input: 1) A word image and 2) a lexicon that always includes the truth of the image.

Output: A list of lexicon words ordered according to their similarity to the truth of the image, as judged by the recognizer.

The recognition process can be outlined as follows. First, the recognizer extracts features from the word image and matches the features against internal word models. Then, based on the matches, lexicon words are assigned with scores or confidence values to indicate how close they are to the truth, ordered accordingly and output by the recognizer. If the truth is ranked at the top of the output, then the recognition is deemed successful. Here an assumption is made that the lexicon always includes the truth,[1] so it should be possible to achieve accuracy rate of 100 percent. Henceforth, the terms "performance" and "accuracy rate" will refer to the rate at which the truth is ranked at the top of the output.

According to the black box view of recognizers, performance depends on three major factors: the recognizer R, the image I, and the lexicon L. Therefore, we can write a performance function $p(R, I, L)$ of three variables to describe such dependence. Before the performance function can be constructed quantitatively, we need to know the quantitative factors that are implied by R, L, and I and how they affect performance. Table 1 gives examples of the factors and their desired values necessary to build a high performance word recognizer. It can be seen in the table that factors like "sensitivity to lexicon size," "word similarity," and "writing

1. This information is not provided to the recognizer to improve its recognition.

TABLE 1
Factors and their Desired Values that Result in High-Performance of Word Recognition

	Factor	Desired value
Recognizer	Ability of distinguishing characters	high
	Sensitivity to lexicon size	low
Lexicon	Size	small
	Word similarity	small
Image	Resolution	high
	Noise/Signal	low
	Writing style	clean

style" cannot be easily expressed in a quantitative way and solving this problem is precisely the thrust of this paper.

A perfect performance model must accommodate all different factors, not just those listed in Table 1. However, our aim is not to predict the exact output for each run of the recognizer. Such a predictor will be the recognizer itself. Instead, our aim is to discover how the factors affect the word recognizer performance statistically, which is meaningful in the context of multiple runs of the recognizer.

For recognizers that build word recognition on top of character recognition, it is possible to break the dependence of word recognition on image quality into two parts: word recognition dependence on character recognition and character recognition dependence on image quality. Therefore, if we can measure character recognition accuracy and discover its relation with word recognition accuracy, image quality does not have to be explicitly measured before a performance model can be derived.

2.2 Word Model Abstraction

An important cause of word recognition difficulty is the similarity between candidate words and it is measured based on the recognizer's inner representation of word models. In fact, approaches to measuring distance between two hidden Markov models (HMMs) have been proposed by researchers using Euclidean distance [20], entropy [21], Bayes probability of error [22], etc. Modeling distances for segmentation based recognizers has been recently studied by the authors [18], [10]. However, for recognizers that deal with character models to generate word hypotheses [1], [4] instead of word models, the method of measuring model distance is yet unexplored because of the difficulty posed by absence of techniques for explicit modeling of words.

In our recent research on lexicon density [18], we have applied regression models on experimental data to discover an approximate linear relationship between recognizer performance and lexicon density. The key issue in defining lexicon density was to measure similarity between lexicon words. Different recognizers have different senses of similarity. For example, a recognizer that does not utilize ascender features may confuse a cursive "l" with a cursive "e" when both of them are written with loops. On the other hand, the same "l" and "e" do not look alike to recognizers that can detect ascenders. Thus, in computing lexicon density, we computed the average model distance between any two word entries using the recognizer's inner representation of word models. For an entry "AVE" in the lexicon, its word model may look like Fig. 2c depending on the actual implementation.

Such a model distance takes the detailed inner workings of recognizers into account and, thus, is potentially quite accurate. However, it is obvious that the computation of model distance, where all pairs of candidate word models are matched, is much more expensive than recognition itself where only one feature sequence extracted from the input is matched against word models. Moreover, the computation completely relies on the recognizer's inner modeling of words, which means one must design completely different algorithms when calculating lexicon density for different recognizers. This is not what we set out to accomplish in this paper. Our goal is to derive the performance prediction model while treating the recognizer as a black box.

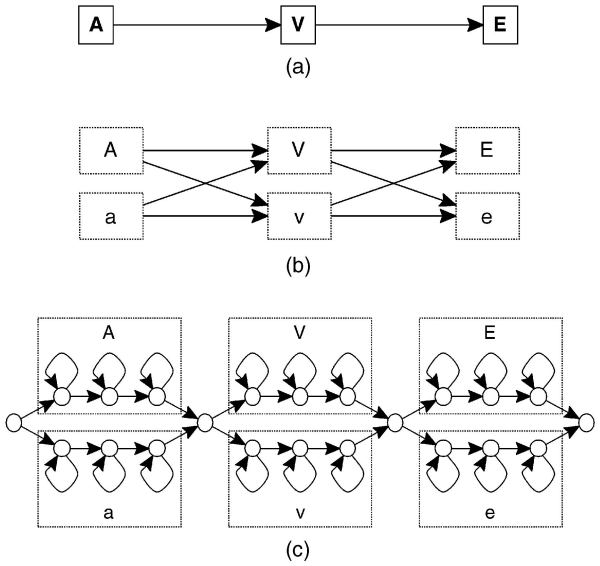


Fig. 2. Word model at different levels of abstraction: (a) case insensitive, (b) case sensitive, and (c) implementation dependent.

Since model distance cannot be easily obtained for different recognizers, we need some other measure of word similarity which is independent of recognizers, easy to compute and accurate. We assume that all word recognizers model words either explicitly or implicitly. Furthermore, we consider a lexicon entry as the abstraction of its word model and obtain two very simple alternatives to word models: one being the case insensitive representation of the lexicon entry and the other being the case sensitive, as illustrated in Fig. 2a and Fig. 2b. We adopt the case insensitive abstraction because of its simplicity, i.e., all words in the lexicon are converted to uppercase and the difference between "Ave" and "Dr" is treated the same way as that between "aVe" and "DR." Under these assumptions, word similarity can be measured by string edit distance which is the minimum number of insertions, deletions, and substitutions to convert one string to another. This measure is independent of recognition methodologies, relatively easy to compute, and accurate.

2.3 Performance Model Derivation

According to the black box view of recognizers as introduced in Section 2.1, the performance function of word recognition is defined as $p(R, I, L)$ where R is the recognizer, I the image, and L the lexicon. R, I, and L can also be viewed as three sets of parameters that characterize the recognizer, the image, and the lexicon, respectively. For the purpose of performance prediction, one would like the function to have the form $p_R(I, L)$ which returns the prediction given an image and a lexicon. However, measuring image quality involves too

many parameters. To simplify, we assume the image quality of training data is representative of testing data and focus on the influence of lexicon. When the parameters related to the recognizer and the image are obtained through a training procedure, the performance function can be rewritten as $p_{R,I}(L)$ and can be used as a predictor of the accuracy rate of recognizer R for a given lexicon, L.

2.3.1 Tournament of Word Candidates

Consider the recognition process as a tournament where nontruths are matched against the truth, where all matches are judged by the recognizer. When a word w_1 wins the match against another word w_2, we say that w_1 *beats* w_2. Obviously, in order for the truth to be ranked at the top, it must beat all other words in the lexicon.

Let us define the *edit distance* between two words as the minimum number of edit operations, including only insertions, deletions, and substitutions to convert one word to the other. When a recognizer is judging the match between the truth and a nontruth, the edit distance between them is provided as the evidence of the truth being the truth and the nontruth being the nontruth. Because the recognizer is not perfect it may ignore parts of the evidence. For example, the edit distance between "l" and "e" is one, but the recognizer may ignore this difference when they are both written with loops. Another example is, when an "l" is written with a long tail, the recognizer may mistakenly take the tail part as an "e" and ignore the difference between "l" and "le." As long as the evidence is not totally ignored, the recognizer can still make the right choice.

Let $t \in L$ be the truth of image I. For an arbitrary nontruth word w, its edit distance to the truth t is denoted by $d(w,t)$. Each of the $d(w,t)$ edit operations is considered as an evidence of t being the truth and w being the nontruth. If the recognizer is aware of at least one such evidence, t wins the match against w. Let q be the probability of one edit operation being ignored by the recognizer ($1-q$ indicates the recognizer's ability of distinguishing characters because edit operations are based on characters) and assume equal importance of all edit operations. Then, the probability that t beats w is $1-q^{d(w,t)}$. In order for t to be the top choice, t needs to beat all $w \in L-\{t\}$. If all matches are independent of each other, then the probability of the truth t being the top choice returned by the recognizer is

$$p_{q,t}(L) = \prod_{w \in L-\{t\}} (1-q^{d(w,t)}). \tag{1}$$

However, the matches are not all independent of each other. The recognizer assigns some distance-based or probability-based score to every candidate. When the truth beats some word w and w beats some other word v, v is not qualified to challenge the truth. That is, transitivity holds for the "beats" relation and we need a new tournament to accommodate such transitivity.

Now consider the recognition process as a *progressive* tournament of word candidates. At the beginning, only one contestant, the truth, participates. Then, other contestants, i.e., other words in the lexicon, are introduced one by one. Unlike the previous tournament in which every contestant is given a chance to challenge the truth, this new tournament qualifies a new contestant to match against the truth only when it is better than all the contestants that have been defeated by the truth. By enforcing this condition, the transitivity of the "beats" relation is maintained. As a result, the expected number of matches against the truth will be fewer than the number of contestants.

2.3.2 Average Number of Matches

Suppose currently the truth t has already defeated a list of random entries F and a new random entry w is added. Notice that w can challenge t only when w is the best in $F \cup \{w\}$. Since all the entries are random, their scores are also random (from some unknown distribution). The chance of w being the best in $F \cup \{w\}$ is $1/|F \cup \{w\}|$.

Let $f(n)$ be the average number of matches against the truth in a lexicon of size n. We have $f(1)=0$ because a lexicon of size one contains only the truth. When $n>1$, the chance of the nth entry challenging the truth is $\frac{1}{n-1}$. Therefore, $f(n)$ can be defined as

$$f(n) = \begin{cases} 0 & , n=1 \\ f(n-1)+\frac{1}{n-1} & , n>1. \end{cases} \tag{2}$$

Thus, $f(n) = 1+\frac{1}{2}+\frac{1}{3}+\ldots+\frac{1}{n-1}$ and $\lim_{n\to\infty} f(n) = \ln(n-1)+\gamma$ where $\gamma = 0.57721\ldots$ is the Euler constant.

The average number of matches gives us an understanding of the tendency of performance change when the lexicon size increases. Since this number is approximately equal to the (natural) logarithm of the lexicon size, it is expected that the performance drop will become less significant as the lexicon size increases, i.e., the performance function might take the form like $(\ldots)^{\ln n}$.

2.3.3 Performance on Lexicon

Let $p(n)$ denote the recognizer's performance on a lexicon of size n. For $n=1$, $p(n)=1$ because a lexicon of size 1 contains only the truth.

In order to derive a simple form of the performance function, we make a simplifying assumption that all nontruths have the same edit distance to the truth and subsequently compensate for this assumption by introducing more model parameters.

When $n>1$, there is $\frac{1}{n-1}$ chance that the nth entry challenges the truth and the probability that the truth wins is $1-q^{\bar{d}(t)}$, where $\bar{d}(t) = \frac{1}{|L|-1}\sum_{w\in L-\{t\}} d(w,t)$ is the average edit distance to the truth. Let $r = q^{\bar{d}(t)}$. The probability that the truth is still at the top after the addition of the nth entry, given that it was at the top after the addition of the $(n-1)$th entry, is $\frac{1}{n-1}(1-r)+\frac{n-2}{n-1} = 1-\frac{r}{n-1}$. Therefore, $p(n)$ can be defined as

$$p(n) = \begin{cases} 1 & , n=1 \\ p(n-1)\left(1-\frac{r}{n-1}\right) & , n>1 \end{cases}. \tag{3}$$

When $n>1$,

$$\begin{aligned} p(n) &= \left(1-\frac{r}{1}\right)\left(1-\frac{r}{2}\right)\ldots\left(1-\frac{r}{n-1}\right) \\ &= \frac{(1-r)(2-r)\ldots(n-1-r)}{(n-1)!}. \end{aligned}$$

The Γ function is a well-known extension of factorial to noninteger values and it has the following properties, $\Gamma(x+1) = x\Gamma(x)$ and $\Gamma(n+1) = n!$, where x is a real number and n is an integer. So, we have

$$\begin{aligned}\Gamma(n-r) &= (n-1-r)\Gamma(n-1-r)\\ &= (n-1-r)(n-2-r)\Gamma(n-2-r)\\ &= \dots\\ &= (n-1-r)(n-2-r)...(1-r)\Gamma(1-r),\end{aligned}$$

which gives us

$$p(n) = \frac{\Gamma(n-r)}{\Gamma(1-r)\Gamma(n)}. \tag{4}$$

We apply the Stirling's asymptotic formula [23]

$$\begin{aligned}\Gamma(x+1) &= \sqrt{2\pi x}\Big(\frac{x}{e}\Big)^x\Big(1+\frac{1}{12x}+\frac{1}{288x^2}-\frac{139}{51840x^3}-\dots\Big)\\ &\simeq \sqrt{2\pi x}\Big(\frac{x}{e}\Big)^x.\end{aligned}$$

for $x \to \infty$ and get

$$\begin{aligned}p(n+1) &\simeq \frac{\sqrt{2\pi(n-r)}(\frac{n-r}{e})^{n-r}}{\sqrt{2\pi n}(\frac{n}{e})^n}\cdot\frac{1}{\Gamma(1-r)}\\ &= \frac{(n-r)^{n-r+1/2}}{n^{n+1/2}}e^r\cdot\frac{1}{\Gamma(1-r)}\\ &= \Big(1-\frac{r}{n}\Big)^{n-r+1/2}e^r n^{-r}\cdot\frac{1}{\Gamma(1-r)}\\ &\simeq n^{-r}\frac{1}{\Gamma(1-r)}.\end{aligned}$$

for $n \to \infty$. Therefore,

$$p(n) \simeq (n-1)^{-r}/\Gamma(1-r) = e^{-r\ln(n-1)+c} \tag{5}$$

for $n \to \infty$ where $c = \ln\frac{1}{\Gamma(1-r)}$.

Equation (5) *asymptotically* reveals the relation between performance and lexicon. However, we are more interested in $p(n)$ when n is relatively small rather than $n \to \infty$. So, $p(n)$ is required to not only meet the initial condition $p(n) = 1$ but also keep its asymptotic form. For this reason, $p(n)$ is estimated as

$$p(n) \simeq e^{-r\ln n}. \tag{6}$$

This new equation replaces $\ln(n-1)$ by $\ln n$ because they are asymptotically the same. $c = \ln\frac{1}{\Gamma(1-r)}$ is ignored because of the initial condition and its closeness to zero.[2]

Thus, after several assumptions, we arrive at $\ln n$ being the approximate number of matches against the truth in a size n lexicon and $(e^{-q^{\bar{d}(t)}})^{\ln n}$ being the approximate performance. It must be pointed out that they are derived when the truth is known, but in the testing environment where predicting performance is more meaningful the truth is never known.

For testing images whose truths are unknown, $\bar{d}(t)$ has to be approximated by the average edit distance between any two entries and the performance function is rewritten as

$$p_q(n, D) = (e^{-q^D})^{\ln n} \tag{7}$$

where $D = \frac{1}{n(n-1)}\sum_{w,v\in L} d(w,v)$ and only one model parameter q present.

2. Typically, the average edit distance $\bar{d}(t)$ is at least two and the probability q is at most 0.9. Correspondingly, c is in the range $[-1.578, 0]$.

Clearly, more parameters have to be introduced to compensate for assumptions and approximations and to keep the model realistic. Based on the above analysis, we conjecture that the performance function has the following form,

$$p_{q,k,a}(n, D) = \Big(e^{-q^D}\Big)^{f(n)} \tag{8}$$

where D is the average edit distance and $f(n) = k\ln^a n$. Here, two new parameters k and a are introduced for the following reasons. First, they do not violate the initial condition that the performance is 100 percent for a lexicon of size one. Second, the model has two degrees of freedom (n and D), but three model parameters are required if the model is to be converted into a multiple regression model. Third, since D approximates $\bar{d}(t)$, the model should be effective at least when D is affinely related to $\bar{d}(t)$.[3]

2.3.4 Multiple Regression Model

The advantage of such a model is that it can be converted to a multiple regression model.

$$\begin{aligned}& p = \Big(e^{-q^D}\Big)^{k\ln^a n}\\ \Leftrightarrow\ & \ln\ p = -q^D k\ln^a n\\ \Leftrightarrow\ & \ln(-\ln p) = D\ln q + a\ln\ln n + \ln k.\end{aligned}$$

Suppose we have a set of observations $\{p_i, n_i, D_i\}$. Let $P_i = \ln(-\ln p_i)$ be the dependent variables, $N_i = \ln\ln n_i$ and D_i be the independent variables, and $\ln q$, a, and $\ln k$ be the regression parameters. We get a multiple regression model

$$P_i = (\ln q)D_i + aN_i + \ln k + e_i = \hat{P}_i + e_i. \tag{9}$$

where $\hat{P}_i$ is the predicted performance and e_i is the residual. Hence, (8) will be referred to as the performance model and (9) as the regression model.

2.3.5 Model Parameters

This performance/regression model takes into account all the performance factors listed in Table 1. First, q is the probability of the recognizer ignoring an edit operation between the truth and a nontruth. This parameter depends on, not only the recognizer, but also the quality of input images because it is less likely that a recognizer will confuse one character with another on high-quality images than on low-quality images. Second, n is the lexicon size and D the similarity between lexicon entries. Third, $f(n) = k\ln^a n$ represents the recognizer's sensitivity to lexicon size.

In character recognition, a misclassification involves one character substitution of the truth by some nontruth. However, in word recognition, a misclassification is the result of a set of character-level edit operations including insertions, deletions and substitutions. Therefore, the parameter q cannot be estimated by the word recognizer's recognition accuracy on characters. It has to be obtained by the regression model. The next section will give details on the experiments for empirically obtaining and verifying model parameters.

3. D is affinely related to $\bar{d}(t)$ if $D = m\bar{d}(t) + l$ for some constants m and l. Section 4 discusses the use of other distance measures instead of edit distance. The same analysis applies here.

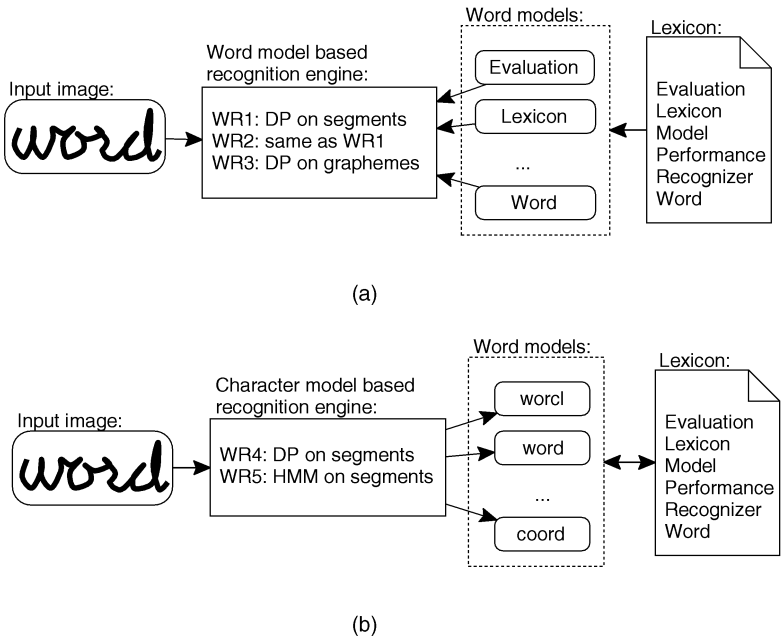


Fig. 3. Strategies of five different word recognizers. (a) WR1, WR2, WR3: Word model based recognition, where the matching happens between the input image and all word models derived from the lexicon; (b) WR4, WR5: Character model based recognition, where the matching occurs between word hypotheses generated by the engine and words in the lexicon.

3 Experiments

3.1 Recognizers

We use five different word recognizers in our experiments.

- WR1: The word recognizer adopts an over-segmentation methodology along with word model based recognition using dynamic programming [5].
- WR2: The recognition methodology is similar to WR1 except for the nature of segmentation and preprocessing algorithms [24].
- WR3: The word recognition methodology is grapheme based and involves no explicit segmentation [25]. It uses word model based recognition with dynamic programming.
- WR4: The word recognizer adopts an over-segmentation methodology along with character model based recognition using dynamic programming [4].
- WR5: The word recognition methodology uses over-segmentation and character model based recognition with continuous density and variable duration hidden Markov models [1].

These five word recognizers can be divided into two categories: word model based recognition and character model based recognition, as illustrated in Fig. 3. In word model based recognition, all lexicon entries are treated as word models and matched against the input. The entry with the best match is the top choice. In character model based recognition, segments are matched against individual characters without using any contextual information implied by the lexicon. Word hypotheses are generated by the character recognition results. If the best hypothesis is found in the lexicon, the recognition is done, otherwise, the second best hypothesis is generated and tested, etc.. Therefore, the lexicon plays an active role in the first strategy but a passive role in the second.

For all the five recognizers, the training phase always results in a set of character models and word models are built on top of character models by concatenation. So it is proper to estimate word recognition accuracy based on character recognition accuracy, as discussed in Section 2.1.

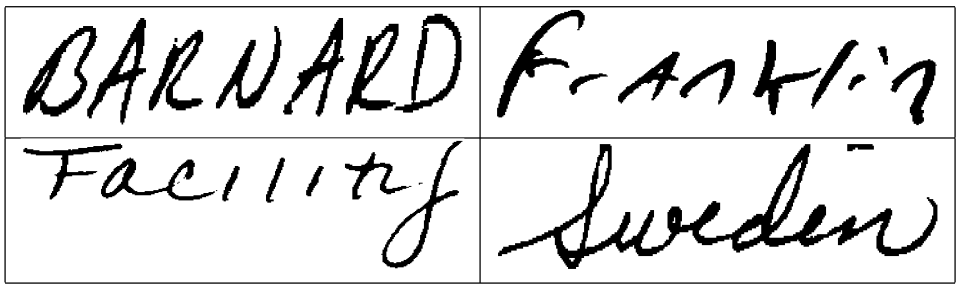


Fig. 4. Example images of unconstrained handwritten words including hand printed, cursive, and mixed.

TABLE 2
Performance Data Collected on Training Set

i	Lex size n_i	Average edit dist D_i	Performance p_i WR1	WR2	WR3	WR4	WR5
1	5	1.834	0.8240	0.8060	0.6273	0.7293	0.8100
2	5	2.137	0.8627	0.8427	0.6727	0.7867	0.8367
3	5	2.414	0.8787	0.8660	0.6920	0.7947	0.8413
4	5	2.708	0.9020	0.8767	0.7453	0.8187	0.8613
5	5	3.169	0.9200	0.8933	0.7520	0.8347	0.8747
6	5	3.556	0.9313	0.9207	0.7920	0.8593	0.9020
7	5	3.915	0.9447	0.9247	0.8233	0.8627	0.9053
8	5	4.263	0.9487	0.9347	0.8473	0.8807	0.9113
9	5	4.668	0.9593	0.9467	0.8580	0.9087	0.9207
10	5	5.248	0.9647	0.9493	0.8953	0.9160	0.9293
11	10	2.193	0.7253	0.7040	0.4367	0.5973	0.7327
12	10	2.429	0.7673	0.7220	0.4920	0.6193	0.7680
13	10	2.678	0.7767	0.7420	0.5093	0.6620	0.7740
14	10	2.938	0.8073	0.7907	0.5567	0.6807	0.7893
15	10	3.533	0.8413	0.8240	0.6160	0.7253	0.8213
16	10	3.867	0.8747	0.8427	0.6573	0.7587	0.8220
17	10	4.232	0.9013	0.8807	0.6920	0.8067	0.8500
18	10	4.538	0.9220	0.9087	0.7420	0.8207	0.8533
19	10	4.867	0.9240	0.9067	0.7613	0.8287	0.8760
20	10	5.329	0.9327	0.9207	0.7900	0.8520	0.8773
21	20	2.426	0.6260	0.5787	0.2987	0.4567	0.6767
22	20	2.605	0.6193	0.5833	0.3353	0.5000	0.6913
23	20	2.843	0.6593	0.6367	0.3620	0.5087	0.7007
24	20	3.041	0.6940	0.6613	0.3787	0.5373	0.7213
25	20	3.750	0.7633	0.7407	0.4860	0.6160	0.7487
26	20	4.028	0.7813	0.7467	0.5093	0.6487	0.7587
27	20	4.431	0.8313	0.8040	0.5827	0.6853	0.7793
28	20	4.687	0.8460	0.8127	0.6053	0.7093	0.7953
29	20	4.982	0.8707	0.8460	0.6493	0.7567	0.8073
30	20	5.344	0.8840	0.8653	0.6667	0.7687	0.8093
31	40	2.571	0.4787	0.4320	0.1760	0.3367	0.6240
32	40	2.698	0.5127	0.4500	0.1987	0.3647	0.6387
33	40	2.955	0.5320	0.4887	0.2020	0.3687	0.6500
34	40	3.110	0.5520	0.5127	0.2093	0.3953	0.6393
35	40	3.887	0.6473	0.6220	0.3433	0.4807	0.6800
36	40	4.101	0.6787	0.6327	0.3567	0.5287	0.6900
37	40	4.568	0.7540	0.7333	0.4267	0.5753	0.7093
38	40	4.783	0.7753	0.7420	0.4853	0.6113	0.7220
39	40	5.068	0.8040	0.7673	0.5113	0.6667	0.7407
40	40	5.347	0.8347	0.7873	0.5520	0.6840	0.7480

3.2 Image Set

All experiments are conducted on a set of 3,000 US postal word images of unconstrained writing styles. All of the images are digitized at 212 dpi. Fig. 4 shows some examples. The 3,000 images are divided into two equal parts, one for training and the other for testing.

3.3 Lexicon Generation

To test the dependence of performance on lexicon size, we generate lexicons of size 5, 10, 20, and 40 for each image. For each lexicon size, 10 lexicons are generated and ordered in ascending order of average edit distance. These 40 lexicons are marked as $L_{j,1}, L_{j,2}, \ldots, L_{j,40}$ for the jth image. In order to allow wide variation of average edit distances, these 40 lexicons actually contain meaningless entries that are random combinations of characters. Besides, three additional lexicons of size 10, 100, and 1,000 are also included as $L_{j,41}, L_{j,42}$ and $L_{j,43}$, respectively. These three lexicons were generated several years ago [26] containing mostly meaningful postal words and they have been used in testing different word recognizers since.

3.4 Determining Model Parameters

We gather performance data on the training set, which contains 1,500 images and 40 lexicons for each image and each word recognizer. In order to get robust estimates of model parameters that can be satisfactorily used on testing data where truths are unknown, we ignore information about truths on the training data as well. Therefore, the average edit distance between any two entries is used instead of that between the truth and other entries. The performance data is collected in Table 2. Notice that D_i is actually the average of average edit distances over 1,500 lexicons, $L_{1,i}, L_{2,i}, \ldots, L_{1500,i}$ for the ith lexicon set. Thus, we have a set of observations $O = \{n_i, D_i, p_i | i = 1 \ldots 40\}$ for each of the five recognizers and regression is performed on this data set.

The multiple regression model is directly applied from (9),

$$P_i = (\ln q)D_i + aN_i + \ln k + e_i = \hat{P}_i + e_i,$$

where $P_i = \ln(-\ln p_i)$ are the dependent variables, D_i and $N_i = \ln \ln n$ are the independent variables, $\ln q$, a, and $\ln k$ are the regression parameters, and $\hat{P}_i$ are the predictions of regression function and e_i are the residuals/errors. The purpose of regression is to minimize the sum of square errors $\sum e_i^2$ for the data in Table 2. Table 3 gives the regression results including parameters, standard errors of the parameters, standard errors of estimate and coefficients of multiple determination.

The standard errors of the parameters are so small that the probability of the null hypothesis $H_0 : \beta = 0$ being true is at most 2×10^{-20}, where β is either $\ln q$, a, or $\ln k$ thus, ensuring that none of the parameters are redundant.

The *Standard Error of Estimate* is defined as $\sigma = \sqrt{\frac{\sum e_i^2}{|O|-3}}$, where $|O|$ is the number of observations and three is the number of parameters in the regression model. Fig. 5 shows two regression planes for WR1 and WR5 (other planes are similar and omitted) to visually illustrate goodness of the

TABLE 3
Regression Parameters Obtained for Five Word Recognizers

Recognizer	$\ln q$	a	$\ln k$	σ	R^2
WR1	-0.4960±0.0101	2.2426±0.0339	-1.9344±0.0453	0.0652	0.9936
WR2	-0.4604±0.0115	2.1278±0.0385	-1.7857±0.0515	0.0741	0.9907
WR3	-0.3966±0.0062	2.0177±0.0208	-1.0328±0.0278	0.0400	0.9968
WR4	-0.3729±0.0077	1.9326±0.0256	-1.4650±0.0342	0.0493	0.9947
WR5	-0.2479±0.0108	1.5142±0.0361	-1.9805±0.0482	0.0694	0.9818

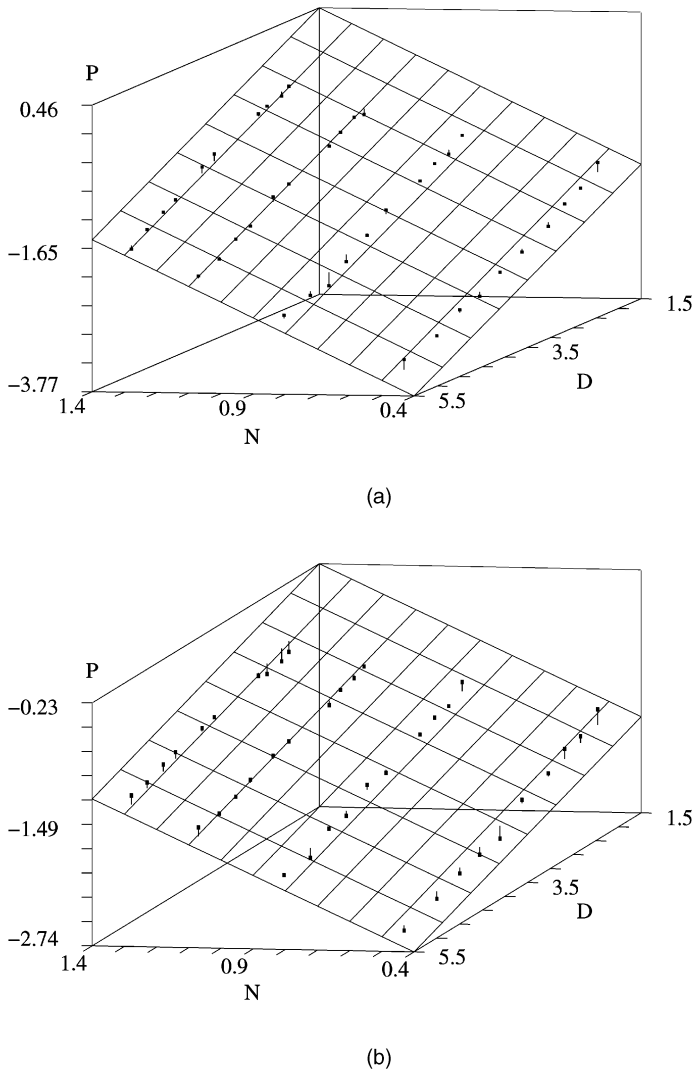


Fig. 5. The regression planes for (a) WR1 and (b) WR5.

fit, where solid dots represent observations and error bars connect observations and predictions.

Coefficient of Multiple Determination is defined as $R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST}$. Here, $SST = \sum(P_i - \bar{P})^2$, where $\bar{P}$ is the average of P_i, and measures the variation in the observed response. $SSR = \sum(\hat{P}_i - \bar{P})^2$ measures the "explained" variation and $SSE = \sum(P_i - \hat{P}_i)^2$ measures the "unexplained" variation. Therefore, R^2 indicates the proportion of variation in the data which is explained by the regression model. A value of $R^2 = 1$ means that the regression model passes through every data point. A value of $R^2 = 0$ means that the model does not describe the data any better than the average of the data. Table 3 shows that about 99 percent of data variation has been explained by the regression model.

The 95 percent confidence intervals of q, a, and k are given in Table 4. In fact, these intervals are calculated based on 95 percent confidence intervals of $\ln q$, a, and $\ln k$. As can be seen, sizes of the intervals are quite small, indicating the robustness of the regression model.

TABLE 4
95 Percent Confidence Intervals of Parameters

	q			a			k		
Recognizer	Value	Lower	Upper	Value	Lower	Upper	Value	Lower	Upper
WR1	0.6089	0.5966	0.6216	2.2426	2.1740	2.3112	0.1445	0.1318	0.1584
WR2	0.6310	0.6165	0.6459	2.1278	2.0498	2.2058	0.1677	0.1511	0.1861
WR3	0.6726	0.6642	0.6811	2.0177	1.9756	2.0598	0.3560	0.3365	0.3766
WR4	0.6887	0.6781	0.6995	1.9326	1.8807	1.9845	0.2311	0.2156	0.2477
WR5	0.7804	0.7636	0.7977	1.5142	1.4411	1.5872	0.1380	0.1252	0.1522

TABLE 5
Performance Data Collected on Testing Set

i	Lex size n_i	Average edit dist D_i	Performance p_i WR1	WR2	WR3	WR4	WR5
1	5	1.847	0.8176	0.7923	0.6237	0.7234	0.8123
2	5	2.143	0.8617	0.8350	0.6738	0.7762	0.8270
3	5	2.417	0.8657	0.8597	0.6939	0.7796	0.8437
4	5	2.701	0.8945	0.8744	0.7306	0.8136	0.8544
5	5	3.155	0.9118	0.9085	0.7614	0.8383	0.8751
6	5	3.509	0.9178	0.9065	0.7948	0.8524	0.8871
7	5	3.872	0.9332	0.9285	0.8115	0.8758	0.9005
8	5	4.247	0.9452	0.9359	0.8436	0.8864	0.9058
9	5	4.650	0.9539	0.9459	0.8656	0.9005	0.9158
10	5	5.243	0.9606	0.9666	0.8984	0.9112	0.9292
11	10	2.192	0.7295	0.7014	0.4579	0.6219	0.7442
12	10	2.431	0.7729	0.7288	0.4786	0.6426	0.7589
13	10	2.685	0.7669	0.7288	0.5154	0.6620	0.7629
14	10	2.936	0.8036	0.7862	0.5675	0.6774	0.7722
15	10	3.511	0.8410	0.8156	0.6277	0.7341	0.7976
16	10	3.842	0.8664	0.8397	0.6638	0.7522	0.8083
17	10	4.216	0.8737	0.8644	0.6892	0.7882	0.8424
18	10	4.520	0.8958	0.8918	0.7253	0.8103	0.8470
19	10	4.862	0.9192	0.9051	0.7587	0.8223	0.8657
20	10	5.311	0.9365	0.9259	0.7794	0.8557	0.8711
21	20	2.424	0.6059	0.5538	0.3008	0.4729	0.6673
22	20	2.598	0.6353	0.5798	0.3229	0.4803	0.6774
23	20	2.841	0.6627	0.6079	0.3616	0.5251	0.7148
24	20	3.036	0.6713	0.6466	0.3783	0.5371	0.7041
25	20	3.754	0.7435	0.7161	0.4846	0.6146	0.7488
26	20	4.020	0.7802	0.7528	0.5114	0.6293	0.7555
27	20	4.415	0.8230	0.7882	0.5916	0.6947	0.7776
28	20	4.673	0.8510	0.8176	0.6116	0.7188	0.7809
29	20	4.960	0.8577	0.8330	0.6477	0.7508	0.7923
30	20	5.306	0.8784	0.8617	0.6918	0.7595	0.8036
31	40	2.573	0.4776	0.4182	0.1832	0.3393	0.5945
32	40	2.699	0.4930	0.4369	0.2045	0.3774	0.6126
33	40	2.958	0.5150	0.4783	0.2126	0.3727	0.6226
34	40	3.103	0.5364	0.4910	0.2340	0.4095	0.6333
35	40	3.894	0.6513	0.5972	0.3329	0.5003	0.6687
36	40	4.089	0.6680	0.6226	0.3603	0.5210	0.6834
37	40	4.549	0.7368	0.6874	0.4418	0.5745	0.7014
38	40	4.758	0.7629	0.7214	0.4786	0.5992	0.7255
39	40	5.031	0.7876	0.7589	0.5261	0.6446	0.7308
40	40	5.308	0.8043	0.7789	0.5441	0.6720	0.7488

3.5 Model Verification

In order to see how the model predicts performance for lexicons other than those included in training, we apply it to the second half of the image set using the parameters obtained from the first half, i.e., parameters in Table 4. Involved lexicons are $L_{j,i}$, $j = 1,501 \ldots 3,000$, and $i = 1 \ldots 43$. The performance data is collected as $\{n_i, D_i, p_i\}, i = 1 \ldots 40$ (Table 5) and $i = 41, 42, 43$ (Table 6).

We use (8) to predict the performance $\hat{p}_i = (e^{-q^{D_i}})^{k \ln^a n_i}$. The results given in Table 6 consist of two parts. The first part is for lexicons $L_{j,1} \ldots L_{j,40}$, where the standard errors of prediction

$$\sqrt{\frac{\sum (p_i - \hat{p}_i)^2}{40}}$$

are given. As can be seen, the model makes only slightly over 1 percent error in its prediction for the five recognizers. Since this part does not contain any lexicon sizes that are beyond the training data, the low prediction errors are expected. The second part is for lexicons $L_{j,41}, L_{j,42}$, and $L_{j,43}$, where the actual performance, the predicted performance, and the

TABLE 6
Verification of the Model on Testing Set

	$L_{i,1} \ldots L_{i,40}$	$L_{i,41}$: n=10, D=6.726			$L_{i,42}$: n=100, D=6.757			$L_{i,43}$: n=1000, D=7.543		
	std. err.	actual	pred.	diff.	actual	pred.	diff.	actual	pred.	diff.
WR1	0.0136	0.9599	0.9672	0.0073	0.8838	0.8560	-0.0278	0.7232	0.7700	0.0468
WR2	0.0163	0.9619	0.9563	-0.0056	0.8631	0.8248	-0.0383	0.6875	0.7277	0.0402
WR3	0.0105	0.8757	0.8755	-0.0002	0.6564	0.5875	-0.0689	0.3929	0.4138	0.0209
WR4	0.0108	0.9092	0.9100	0.0008	0.7856	0.7006	-0.0850	0.5906	0.5593	-0.0313
WR5	0.0122	0.9118	0.9120	0.0002	0.8013	0.7703	-0.0310	-	0.6724	-

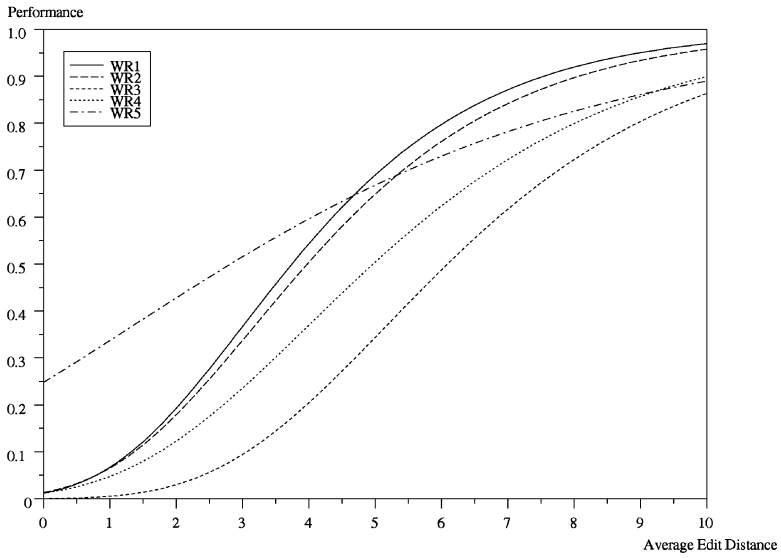


Fig. 6. Typical performance curves when lexicon size is 100.

difference between them are given for each lexicon and each recognizer,[4] the prediction errors for lexicon size 10 are very small as expected. The errors for lexicon sizes 100 and 1,000 are larger but less than 0.045 on average. Therefore, notwithstanding the larger prediction errors, the performance model still generalizes itself to larger lexicons and larger average edit distances.

4 Discussions

Comparison of Recognizers: Some interesting traits of the recognizers can be observed by analysis of the three model parameters. First, the q parameter is the probability of a recognizer ignoring one edit operation between truth and nontruth. In other words, smaller q means higher ability of distinguishing characters. So, based on the values of q, we say WR1 is the best among the five in distinguishing characters in words. Moreover, larger q also means smaller improvement in accuracy when average edit distance increases, that is exactly what Table 2 shows for WR5. Second, a and k together, indicate a recognizer's sensitivity to the change in lexicon size while a is in terms of order of magnitude and k is in terms of coefficiency. In this sense, WR5 is the least sensitive and its performance drop is the least when lexicon size increases, as shown in Table 2. Fig. 6 shows a set of typical performance curves when lexicon size is 100. WR1 is undoubtedly the best among WR1, WR2, WR3, and WR4, while WR5 is better than WR1 when average edit distance is below 4.5. Therefore, to summarize, WR1 and WR5 can be considered to be the best recognizers among the five. WR1 is superior when lexicon entries are very different. WR5 is quite insensitive to the change in lexicon size and is especially good for difficult recognition tasks when lexicon size is large and lexicon entries are similar.

Influence of Word Length: Grandidier et al. [13] have reported that the influence of word length on recognition has two aspects. First, long words are easier to recognize than short words. Second, lexicons consisting of long words are easier than those consisting of short words. According to our performance model, larger average edit distance implies higher performance. This supports both the aspects of the influence of word length simply by the fact that the average edit distance to a long word is generally higher than that to a short word. When the long word is the truth, other words tend to be far from it in terms of edit distance. When the long word is in the lexicon but not the truth, the truth also tends to be far from it for the same reason. We illustrate this by Fig. 7, where performance data is collected on $L_{i,41}$, lexicons of size 10. The lexicons are divided into three groups, each containing about 1,000 lexicons. These three groups are representing short truths (two-four characters), medium truths (five-seven characters), and long truths (eight and above), and their average distances are 6.205, 6.816, and 7.205, respectively. The recognition rates of the five recognizers are given as bars and the predictions are given as curves. Generally, recognizers

4. The data on lexicon size 1,000 for WR5 is not available because it is incapable of handling such large lexicon size without modifying the source code.

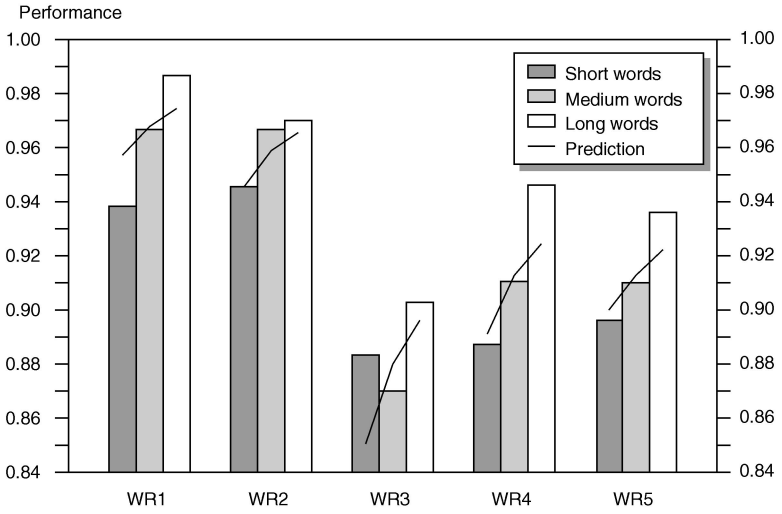


Fig. 7. Influence of word length explained by the performance model where the average edit distances are 6.205, 6.816, and 7.205 for short words, medium words, and long words, respectively.

perform better on long words than on short words because long words have higher average edit distances than short words. The predictions can be seen as being quite close to the actual numbers.

Using Other Distance Measures: As discussed in Section 2.2, the popularity of edit distance is because of its simplicity and independence from recognizers. Nevertheless, questions may arise when there is some other distance measure available, such as model distance[5] in calculating lexicon density [18]. One may enquire how model distances are related to edit distance in predicting performance.

When some model distance D_M is affinely related to edit distance D, i.e., $D = mD_M + l$, the performance model $(e^{-q^D})^{k\ln^a n}$ from (8) can be rewritten as

$$\left(e^{-q^{mD_M+l}}\right)^{k\ln^a n} = \left(e^{-(q^m)^{D_M}}\right)^{kq^l \ln^a n}, \tag{10}$$

taking the same form as (8) by replacing q^m with q' and kq^l with k'. That is, the performance model can be directly applied to any distance measure that is affinely related to edit distance.

To support the above conclusion, we apply the performance model on data previously collected in [18] and use recognizer-dependent model distance instead of edit distance. Because the calculation of model distance completely relies on the implementation of word recognizers and involves heavy computation, only data for WR1 and WR3 is available in [18]. Fig. 8 shows that model distance defined for WR1 (scaled up four times for better observation) is almost affinely related to edit distance but this is not so for WR3. We obtain the standard errors of prediction in Table 7. As can be seen, the use of model distance is only marginally better than edit distance and the performance model we have proposed in this paper is more accurate than the approach in [18]. The exception in case of WR3 can be explained by the fact that the model distance for WR3 is not affinely related to the edit distance.

5. Called "slice distance" for WR1 and "grapheme distance" for WR3 in [18].

5 Conclusions and Future Work

In this paper, we investigate the dependence of word recognition on lexicons and propose a quantitative model to directly associate the performance of word recognizers with lexicon size and the average edit distance between lexicon entries. The proposed model has three model parameters q, k, and a, where q captures the recognizer's ability to distinguish characters, and $f(n) = k\ln^a n$ captures the recognizer's sensitivity to a lexicon size n. This model emphasizes the effect of lexicons and does not have any explicit parameters to measure image quality. However, it decomposes the

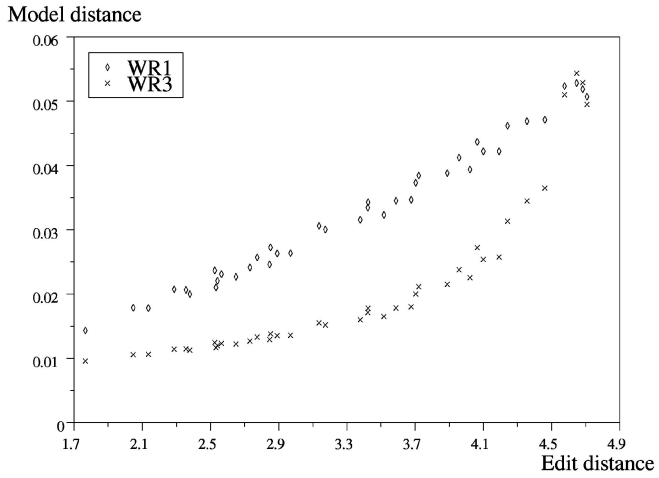


Fig. 8. Edit distance versus model distance for WR1 and WR3.

TABLE 7
Comparison of Standard Errors in Prediction Using Model Distance and Edit Distance

	Result from [18]		Result from Equation 8	
Recognizer	Model dist.	Edit dist.	Model dist.	Edit dist.
WR1	0.0157	0.0216	0.0078	0.0080
WR3	0.0190	0.0225	0.0565	0.0099

dependence of word recognition on image quality into two parts: word recognition on character recognition and character recognition on image quality, where the first part is embodied in the form of the model and the second part in the parameter q. We use synthetic lexicons to get performance data on five different word recognizers and then use multiple regression to derive the model parameters. Statistical analysis is shown to strongly support the model.

The model is derived, based on the assumption that word recognition is a combination of character recognition results, hence, it can be generalized to all word recognizers that model characters. Experimental results on five different recognizers have shown the generality of this model. However, for recognizers that model words as whole without identifying individual characters, it is still unknown if the model is feasible.

The availability of such a model not only helps in understanding a recognizer's behavior but also promises applications in improving word recognition by predicting performance. Once the performance of recognizers can be predicted, the prediction can be used in selecting and combining recognizers. For example, observing different performance curves such as those in Fig. 6, we are able to decide which recognizer to use or with what weights to combine them when the lexicon changes.

The proposed performance model has the form $p_{R,I}(L)$, which means variables related to the lexicon L can be freely supplied while parameters derived from the recognizer R and the training image set I must be fixed. This seems to be a little inconvenient because what we actually want is the form $p_R(I, L)$ to allow the adaption of performance prediction to both the image and the lexicon. Moreover, since the model works only for top choice accuracy rates, a more challenging task will be finding a generalized model that is capable of predicting top N choices accuracy rates. These will be considered in the future.

REFERENCES

[1] M. Chen, A. Kundu, and S. Srihari, "Variable Duration Hidden Markov Model and Morphological Segmentation for Handwritten Word Recognition," *IEEE Trans. Image Processing,* vol. 4, pp. 1675-1688, Dec. 1995.

[2] G. Dzuba, A. Filatov, D. Gershuny, and I. Kil, "Handwritten Word Recognition—The Approach Proved by Practice," *Proc. Sixth Int'l Workshop Frontiers in Handwriting Recognition,* pp. 99-111, 1998.

[3] 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 Trans. Pattern Analysis and Machine Intelligence,* vol. 21, pp. 752-760, Aug. 1999.

[4] J. Favata, "Character Model Word Recognition," *Proc. Fifth Int'l Workshop Frontiers in Handwriting Recognition,* pp. 437-440, Sept. 1996.

[5] G. Kim and V. Govindaraju, "A Lexicon Driven Approach to Handwritten Word Recognition for Real-Time Applications," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, no. 4, pp. 366-379, 1997.

[6] M. Mohammed and P. Gader, "Handwritten Word Recognition Using Segmentation-Free Hidden Markov Modeling and Segmentation-Based Dynamic Programming Techniques," *IEEE Trans Pattern Analysis and Machine Intelligence,* vol. 18, pp. 548-554, May 1996.

[7] U. Marti and H. Bunke, "On the Influence of Vocabulary Size and Language Models in Unconstrainted Handwritten Text Recognition," *Proc. Sixth Int'l Conf. Document Analysis and Recognition,* pp. 260-265, Sept. 2001.

[8] J. Park and V. Govindaraju, "Using Lexical Similarity in Handwritten Word Recognition," *IEEE Conf. Computer Vision and Pattern Recognition,* 2000.

[9] G. Seni, V. Kripasundar, and R. Srihari, "Generalizing Edit Distance to Incorporate Domain Information," *Pattern Recognition,* vol. 29, no. 3, pp. 405-414, 1996.

[10] P. Slavik and V. Govindaraju, "Use of Lexicon Density in Evaluating Word Recognizers," *Multiple Classifier Systems,* pp. 310-319, June 2000.

[11] L.R. Bahl, F. Jelinek, and R.L. Mercer, "A Maximum Likelihood Approach to Continuous Speech Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 5, Mar. 1983.

[12] *Survey of the State of the Art in Human Language Technology,* R.A. Cole, J. Mariani, H. Uszkoreit, A. Zaenen, and V. Zue, eds., Cambridge Univ. Press, 1998.

[13] F. Grandidier, R. Sabourin, A. E. Yacoubi, M. Gilloux, and C.Y. Suen, "Influence of Word Length on Handwriting Recognition," *Proc. Fifth Int'l Conf. Document Analysis and Recognition,* pp. 777-780, Sept. 1999.

[14] H.S. Baird, *Structured Document Image Analysis, Document Image Defect Models* pp. 546-556, Springer-Verlag, 1992.

[15] H.S. Baird, "State of the Art of Document Image Degradation Modeling," *IAPR Workshop Document Analysis Systems,* Dec. 2000.

[16] T.K. Ho and H.S. Baird, "Evaluation of OCR Accuracy Using Synthetic Data," *Proc. Third Int'l Conf. Document Analysis and Recognition,* pp. 278-282, Aug. 1995.

[17] T.K. Ho and H.S. Baird, "Large-Scale Similation Studies in Image Pattern Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, pp. 1067-1079, Oct. 1997.

[18] V. Govindaraju, P. Slavik, and H. Xue, "Use of Lexicon Density in Evaluating Word Recognizers," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 6, pp. 789-800, June 2002.

[19] V. Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions and Reversals," *Soviet Physics–Doklady,* vol. 10, no. 8, pp. 707-710, 1966.

[20] S. Levinson, L. Rabiner, and M. Sondhi, "An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition," *AT&T Technical J.,* vol. 62, no. 4, pp. 1035-1074, 1983.

[21] B. Juang and L. Rabiner, "A Probabilistic Distance Measure for Hidden Markov Models," *AT&T Tech. J.,* vol. 64, no. 2, pp. 391-408, 1985.

[22] C. Bahlmann and H. Burkhardt, "Measuring HMM Similarity with the Bayes Probability of Error and Its Application to Online Handwriting Recognition," *Proc. Sixth Int'l Conf. Document Analysis and Recognition,* pp. 406-411, 2001.

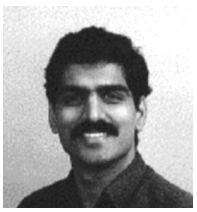
[23] M. Abramowtiz and I. Stegun, *Handbook of Mathematical Functions.* New York: Dover, 1964.

[24] P. Slavik and V. Govindaraju, "An Overview of Run-Length Encoding of Handwritten Word Images," Technical Report 09, State Univ. at New York at Buffalo, Aug. 2000.

[25] H. Xue and V. Govindaraju, "Building Skeletal Graphs for Structural Feature Extraction on Handwriting Images," *Proc. Sixth Int'l Conf. Document Analysis and Recognition,* Sept. 2001.

[26] M. Chen, "Handwritten Word Recognition Using Hidden Markov Models." PhD thesis, State Univ. of NY at Buffalo, Sept. 1993.

Hanhong Xue received the BS and MS degrees in computer science from the University of Science and Technology of China in 1995 and 1998, respectively, and the PhD degree in computer science and engineering from the State University of New York at Buffalo, in June 2002. His research interests include image processing, pattern recognition, and computer vision. He is a student member of the IEEE.



Venu Govindaraju is the author of 130 papers in various journals and conferences in the areas of handwriting recognition and document image analysis. He is the associate editor of the *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, the IEEE Transactions on Systems, Man, and Cybernetics (SMC-B), and the *Pattern Recognition Journal*. He is the program cochair of the Eighth Internatonal Workshop on Frontiers in Handwriting Recogniton to be held at Niagara-on-the-Lake in September 2002. He is also a member of the executive board of the International Graphonomics Society. He is a senior member of the IEEE.