Open access • Proceedings Article • DOI:10.1109/CVPR.2000.854813

Using lexical similarity in handwritten word recognition — Source link

Jaehwa Park, Venu Govindaraju

Institutions: State University of New York System

Published on: 13 Jun 2000 - Computer Vision and Pattern Recognition

Topics: Lexical similarity, Intelligent word recognition, Lexicon, Intelligent character recognition and Word recognition

Using Lexical Similarity in Handwritten Word Recognition

Jaehwa Park and Venu Govindaraju
Center of Excellence for Document Analysis and Recognition (CEDAR)
Department of Computer Science and Engineering
State University of New York at Buffalo
Amherst, NY 14260

Abstract

Recognition using only visual evidence cannot always be successful due to limitations of information and resources available during training. Considering relation among lexicon entries is sometimes useful for decision making. In this paper, we present a method to capture lexical similarity of a lexicon and reliability of a character recognizer which serve to capture the dynamism of the environment. A parameter, lexical similarity, is defined by measuring these two factors as edit distance between lexicon entries and separability of each character's recognition results. Our experiments show that a utility function considering lexical similarity in a decision stage can enhance the performance of a conventional word recognizer.

1 Introduction

Shapes of characters in handwriting vary widely. Some of the variation in the shpae of handwriting are categorized and methods to normalize the variations have been developed in previous literature [1]. However, still the variation of character shapes is one of the major obstacles in handwriting recognition reaching the high recognition performance seen in ther recognition of machine printed documents.

Usually the target of handwritten word recognition is to choose one word from the dictionary as the final answer. If a lexicon is given, the recognition engine finds the best lexicon entry which has the highest matching score to the array of image segments of the input image or extracted feature array from the image. Scoring is performed between shape based feature vectors extracted from segmented character images and reference prototypes of codewords.

Empirically speaking, recognition using only visual evidence, *i.e.* distance between shape based feature vector extracted from an input image and reference prototypes, cannot always be successful due to the limitations of information and resources available during training. If a word recognition engine is operated on shape based features only, the success of finding the true answer with high confidence values can not always be achieved, because some of the characters in a word sometimes have poor shapes, as seen on the images of the last line in Figure 1.

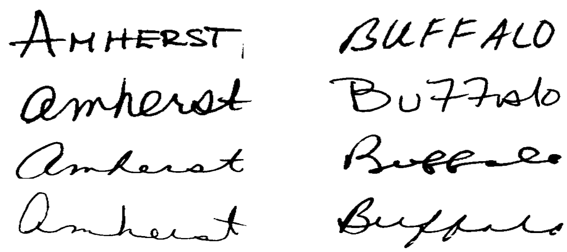


Figure 1. Handwritten words in different writing style

However, "Amherst" and "Buffalo" can still be recognized, if the characters whose images have sharp shape features are unique in the lexicon (for example, based on character recognition of 'A','t','B' and 'f' of the images). Additional lexical similarity measurement which describes uniqueness of characters in a lexicon for the purpose of accepting the lexicon entry as a valid answer, must be considered to reach higher performance. In more general words, capturing the decision environment of the recognizer is required to overcome the incompleteness of shape based recognition.

In this paper, we present a method to capture the dynamism of the decision environment represented by the class (character) similarity of a lexicon and reliability of the character recognizer.

2 Methodology

A lexicon driven word recognizer finds the best matching word in a given lexicon. In analytical approaches, word recognition engines find the best matches between lexicon entries and an array of pre-segmented strokes of word images (Figure 2). Comparisons are made between shape oriented feature vectors extracted from possible combinations of segments and reference prototypes of codewords to generate matching scores. A global optimum path is obtained for each lexicon entry and corresponding confidence values of the path is provided as a recognition result [2].

If shape features from an input image are very sharp, the image is recognized easily as the true answer presented in a given lexicon (images in the first line of Figure 1). However, if shape features are not so close to prototypes as the images shown in the last line of Figure 1, more attention is required to accept the true words, compared to the images above.

For instance, the image of *Amherst* in the last line can be recognized when there is no ambiguity in the position of *r* in the given lexicon since the visual evidence of isolated character image at position *r* does not much match the general shape of *r*. If the lexicon entries are given as two possible choices of *Amherst* and *Buffalo*, we can easily choose *Amherst* as an answer based on strong shape oriented recognition evidence of *A, h* and *t*. If the lexicon is given as *Amherst* and *Amheist*, we can not choose *Amherst* as the final answer.

2.1 Similarity Measure

Lexical similarity measurement (in distance) between two finite character arrays can be measured by *Edit Distance* [3]. Given two strings, the edit distance is defined as the minimum cost of transforming one string to the other through a sequence of weighted edit operations. The edit operations are usually defined in terms of *insertion*, *deletion*, and *substitution* of symbols.

However in the lexical similarity measurement especially in segmentation driven word recognition, the simple cost model between characters, such as confusion tables of character recognizer, is not accurate enough to express the actual cost of character transformation because each character recognition is performed on a set of grouped segmented stroke array called *segment frame*.

The similarity measurement of a lexicon entry depends not only on the lexicon but also on the separation characteristics of the character recognizer. If the recognizer being employed has high inter-class confusion for particular character pairs, the similarity (from the point of recognition separation of entries) which have the confusion pairs in their character arrays is very high.

We extend the edit distance by adding three edit operations; *splitting*, *merging* and *converting* in order to let it work in a segment frame. The edit distance is weighted by inter-class confusion of a character recognizer since inter-class confusion reflects imperfections of the character recognizer.

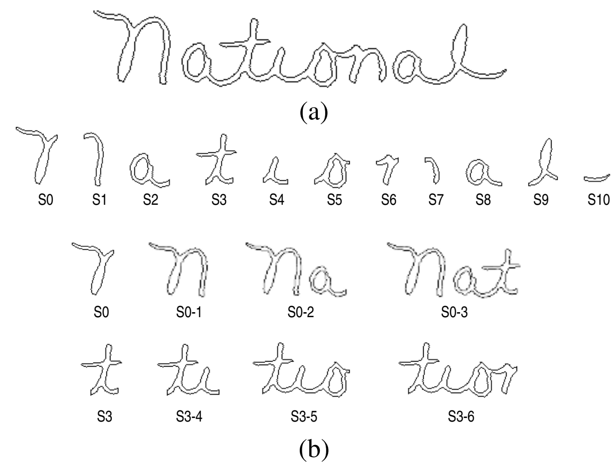


Figure 2. example of stroke segmentation array; (a) contour representation of input image and (b) segmented strokes and combination of subset of strokes

3 Framework

3.1 Recognition Confidence

Consider a lexicon, H, which has N_H lexicon entries and a lexicon entry, $\mathbf{h}_i$, over a finite alphabet, of length N_{h_i}, given as $H = \{\mathbf{h}_0, \mathbf{h}_1, \cdots, \mathbf{h}_{N_H-1}\}$, and $\mathbf{h}_i = [h_{i,0}, h_{i,1}, \cdots, h_{i,N_{h_i}-1}]$, where the class set C is a possible character set including the null ϕ, $C = \{\phi, c_1, c_2, \cdots, c_{N_c}\}$, and $h_{i,j} \subset C$. Simply put, $\mathbf{h}_i$ is a word, and $h_{i,j}$ is the j^{th} character of the word $\mathbf{h}_i$. If upper and lower case alphabets or similar shape characters, (0,O) (1,I,l), (2,Z), etc, are allowed to be folded as possible combinations of a lexicon word, $h_{i,j}$ can be a subset that includes all possible character at the position.

Let us assume that a stroke segmentation algorithm is applied to a given image and a stroke segment array $\mathbf{s} = [s_0, s_1, \cdots, s_{N_s-1}]$ of length N_s is generated, where s_i is a stroke segment. An example is shown in Figure 2.

A dynamic programming based word recognizer similar to [2] finds the best matching path of a lexicon entry within a segment array conforming to rules such as (i) at least one character matches one stroke segment, (ii) one character can only match within a window, and (iii) lexicon entries with a length smaller than N_s or the maximum allowable number of characters of stroke array are discarded.

The confidence of any h associated with sub-array between stroke segment s_a and s_b of $\mathbf{s}$ is given by

$$A^{a,b}(h) = \min_{\forall c \in h} \delta_h(c) \qquad (1)$$

where $\delta_h(c)$ is the confidence measure function of class c in pre-built prototypes of a character recognition engine (in our implementation it is in distance measure metric). Recognition confidence of a lexicon entry $\mathbf{h}$ is given by a minimum distance matching array of the individual class subset,

$$A(\mathbf{h}) = [A^{a_0,b_0}(h_0), \cdots, A^{a_{N_h-1},b_{N_h-1}}(h_{N_h-1})] \tag{2}$$

where $a_0 = 0$, $b_{N_h-1} = N_s - 1$ and $a_{i+1} - b_i = 1$.

The conventional recognition confidence of a lexicon entry is given by the average of character recognition confidence in the lexicon entry as,

$$|A(\mathbf{h})| = \frac{1}{N_h} \sum_{i=0}^{N_h-1} A^{a_i,b_i}(h_i) \tag{3}$$

3.2 Edit Cost in Segment F rame

We can define the edit operations between the best matching paths of two different lexicon entries, $\mathbf{h}_i$ and $\mathbf{h}_j$, without altering the segmented stroke frame of matching as follows

- *nullifying*: $r(h_{i,m} \to \phi | s_{a\sim b})$
- *substitution*: $r(h_{i,m} \to h_{j,n} | s_{a\sim b})$
- *splitting*: $r(h_{i,m} \to [h_{j,n_1}, \cdots, h_{j,n_2}] | s_{a\sim b})$
- *merging*: $r([h_{i,m_1}, \cdots, h_{i,m_2}] \to h_{j,n} | s_{a\sim b})$
- *converting*: $r([h_{i,m_1}, \cdots, h_{i,m_2}] \to [h_{j,n_1}, \cdots, h_{j,n_2}] | s_{a\sim b})$

where $r(h_{i,m} \to h_{j,n} | s_{a\sim b})$ is a transform function from a symbol subset $h_{i,m}$ to $h_{j,n}$ within stroke segments array of s_a to s_b. m and n denote indices of characters within a lexicon entry, *i.e.* $h_{i,m}$ denotes the m^{th} character in lexicon entry i where $h_{i,m} \subset C$. a and b are indices of the stroke segment array, and $s_{a\sim b}$ denotes any character combination of stroke segment s_a through s_b where $0 \le a \le b \le N_s - 1$.

Nullifying is a transform operation for a character match a certain boundary of stroke segments to null. *Substitution* is the replacement of a character match to another within the exact same matching bound. *Splitting* is a split of a character match into several character matches within the segment frame and *merging* is the inverse operation of splitting. *Converting* is transforming a string of multiple character matches into another string matches when matching bounds of characters in the two strings cross each other within the segment frame.

Let $\varepsilon(c_m \to c_n | s_{a\sim b})$ be a elementary transform cost function from c_m to c_n and $\delta_h(c|s_{a\sim b})$ be a character recognition confidence function of c within stroke segments s_a and s_b. The transform cost functions are defined as

$$\varepsilon(h_{i,m} \to \phi | s_{a\sim b}) = \delta_h(h_{i,m} | s_{a\sim b}) \tag{4}$$

$$\varepsilon(h_{i,m} \to h_{j,n} | s_{a\sim b}) = \delta_h(h_{i,m} | s_{a\sim b}) - \delta_h(h_{j,n} | s_{a\sim b}) \tag{5}$$

$$\begin{aligned}\varepsilon(h_{i,m} \to [h_{j,n_1}, \cdots, h_{j,n_2}] | s_{a\sim b}) = \delta_h(h_{i,m} | s_{a\sim b}) - \\ min(\delta_h(h_{j,n_1} | s_{a\sim a_1}) + \delta_h(h_{j,n_1+1} | s_{(a_1+1)\sim a_2}) + \\ \cdots + \delta_h(h_{j,n_2} | s_{a_{n_2-n_1-1}+1\sim b}))\end{aligned} \tag{6}$$

$$\begin{aligned}\varepsilon([h_{i,m_1}, \cdots, h_{i,m_2}] \to h_{j,n} | s_{a\sim b}) = \\ min(\delta_h(h_{j,m_1} | s_{a\sim a_1}) + \delta_h(h_{j,m_1+1} | s_{(a_1+1)\sim a_2}) + \\ \cdots + \delta_h(h_{j,m_2} | s_{a_{m_2-m_1-1}+1\sim b})) - \delta_h(h_{j,n} | s_{a\sim b})\end{aligned} \tag{7}$$

$$\begin{aligned}\varepsilon([h_{i,m_1}, \cdots, h_{i,m_2}] \to [h_{j,n_1}, \cdots, h_{j,n_2}] | s_{a\sim b}) = \\ min(\delta_h(h_{j,m_1} | s_{a\sim a_1}) + \delta_h(h_{j,m_1+1} | s_{(a_1+1)\sim a_2}) + \\ \cdots + \delta_h(h_{j,m_2} | s_{a_{m_2-m_1-1}+1\sim b})) - \\ min(\delta_h(h_{j,n_1} | s_{a\sim a_1}) + \delta_h(h_{j,n_1+1} | s_{(a_1+1)\sim a_2}) + \\ \cdots + \delta_h(h_{j,n_2} | s_{a_{n_2-n_1-1}+1\sim b}))\end{aligned} \tag{8}$$

where $a_1, a_2, \cdots, a_{n_2-n_1-1}$ are indices of stroke segments within $s_a \sim s_b$ satisfying $a \le a_1 \le \cdots a_{n_2-n_1-1} \le b$.

3.3 Matching Distance

A matching transform path from $\mathbf{h}_i$ to $\mathbf{h}_j$, has been defined directly in terms of transform operation of the paths (or traces) $\mathbf{r}_{i,j} = [r_{i,j,0}, r_{i,j,1}, \cdots, r_{i,j,N_{h_i}-1}]$, where $\mathbf{r}_{i,j}$ is a path of transform operations of symbols from $\mathbf{h}_i$ to $\mathbf{h}_j$ within a segment frame $\mathbf{s}$. The associated transform cost for a path is

$$\lambda(\mathbf{r}_{i,j}) = \sum_{k=0}^{N_{h_i}-1} \varepsilon(r_{i,j,k}) \tag{9}$$

where k is a index of matching bound array of the path, $k \in \{(s_{a_0\sim b_0}), \cdots, (s_{a_{N_{h_i}-1}\sim b_{N_{h_i}-1}})\}$.

Let $\bar{\mathbf{r}}_{i,j}$ be the transform path of minimum cost from $\mathbf{h}_i$ to $\mathbf{h}_j$. Then the matching distance of $\mathbf{h}_i$ to $\mathbf{h}_j$ is defined as

$$\lambda(\bar{\mathbf{r}}_{i,j}) = min(\lambda(\mathbf{r}_{i,j})) \tag{10}$$

The overall transform cost of a character of a lexicon entry is defined by the average of minimum transform costs of each character to all corresponding characters of the other lexicon entries in the lexicon.

The overall matching distance of a true character matched in a portion of a segment array can be minimized in negative value, if corresponding characters of other lexicon entries matched to the same segment portion are all different and a character recognition engine generates maximum separation between them.

Otherwise, if most of corresponding characters of other lexicon entries in the same portion are the same true class or if the classifier does not generate good separation between them, the overall matching distance increases, or as a non true class it became a positive value. Thus, the overall matching distance reflects the importance of a character in a lexicon as well as the separation degree provided by a recognition engine.

Lexical similarity of a lexicon entry is defined by an overall transform distance array of the character string of the lexicon entry to all the others in the lexicon. The liability of a character in a lexicon $L(\mathbf{h}_i|H)$ can be evaluated by the lexical similarity.

$$L(\mathbf{h}_i|H) = -\frac{1}{N_H - 1} \sum_{j=0, j\neq i}^{j=N_H-1} \varepsilon(\bar{\mathbf{r}}_{i,j}) \tag{11}$$

The absolute recognition distance of a lexicon entry in Equation 3 is the minimum distance measurement between feature vector and templates of the best path within the stroke segment array. This lexicon complexity of a lexicon entry reflects the lexical (or character) entropy of a lexicon entry projected into a lexicon in terms of matching distance. The overall lexicon complexity is

$$|L(\mathbf{h}_i|H)| = \frac{1}{N_H - 1} \sum_{j=0, j\neq i}^{j=N_H-1} \lambda(\bar{\mathbf{r}}_{i,j}) \tag{12}$$

3.4 Recognition Utility

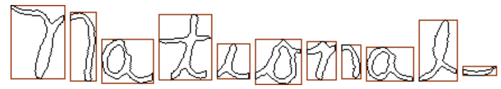
The recognition utility function $U(\mathbf{h}|H)$ of each lexicon entry $\mathbf{h}$ in a lexicon H is defined by a convex combination of recognition confidence A and lexical similarity L such as

$$U(\mathbf{h}|H) = A(\mathbf{h}) - \beta L(\mathbf{h}|H) \tag{13}$$

where $U(\mathbf{h}|H) = [U(h_0|H), \cdots, U(h_{N_h-1}|H)]$. The parameter β is referred to the *index of rejectivity*. Large values of β imply an increased willingness to reject hypotheses. $\beta \cong 0$ implies minimal concern for liability, $\beta \cong 1$ implies equal concern for recognition confidence and lexical similarity [4].

If we set $|U|$ to be a function which generates an overall recognition utility value of a lexicon entry $\mathbf{h}$ as the average value of each recognition utility, then $|U|$ becomes

$$|U(\mathbf{h}|H)| = \frac{1}{N_h} \sum_{i=0}^{N_h-1} U(h_i|H) \tag{14}$$



(a)

(b)

Figure 3. Stroke segmentation: (a) segmented primitive array and (b) prime stroke period estimation

If we set β as zero, $|U|$ becomes a conventional method of finding the average distance between a lexicon entry and a given image using a character recognizer which provides the distance in feature space.

4 Implementation

A segmentation based word recognizer is chosen as the base implementation model of our approach. After preprocessing, a character segmentation algorithm is applied to a given word image and a character recognizer is applied to each segment assuming that perfect character separation is achieved. However since perfect character segmentation is not always guaranteed, an over-segmentation strategy (called as *stroke segmentation*) is usually adopted to avoid character segmentation errors which feed forward to character recognizer.

After stroke segmentation procedure, possible combinations of segmented strokes are sent to a character recognizer to find matching scores of characters, and these scores are used as a metric in next word matching. Various word matching schemes have been implemented according to the metric used in a character classification and the method of handling a lexicon.

In our implementation, a matching scheme using Dynamic Programming and a distance metric are used to find the best match between a symbol string and segmented strokes. Our implementation follows the flow of segmentation and recognition after preprocessing.

4.1 Stroke segmentation

In the segmentation stage, word components are split into more detailed symbolic primitives using the method described in [5]. An example of segmentation results is shown in Figure 2. Then possible combinations of stroke primitives are sent to a character recognizer.

1063-6919/00 $10.00 © 2000 IEEE

A *prime stroke analysis*, which have been introduced in [5] is used as a basic parameter provider. A dominant stroke interval period, *prime spatial period* is estimated in a given word image and used as a reference for choosing a character matching window size. For computational efficiency, a permissible window size for character matching is generated by a function of the number of characters in a lexicon, the number of prime segments in a given image, and extracted prime stroke period from the image.

4.2 F eatures

Features are directly generated from a pixel based contour representation of a character [6]. A given image is divided into N_f by N_f grids in equal area where N_f is the size of divisions. A N_g slots histogram of gradient of contour components are obtained at each grid and an array of the histogram measurements of all grids are used as a feature vector. Two global features, aspect ratio of bounding box and a vertical/horizontal component ratio on overall contours are included in the feature vector in order to prevent mis-classification of non-character components. A $(N_f \cdot N_f \cdot N_g + 2)$ dimensional feature vector is extracted for each image.

In order to generate prototype templates, feature vectors are extracted using a training character set and the same feature vector extraction method. And the extracted feature vectors which belong to a class are clustered independently, using a K-means clustering algorithm. A clustering procedure is completed if the mean square error of clusters to the centers is less than a predetermined threshold or the number of centers reaches the maximum allowable number of clusters.

4.3 Matching

The major goal of character matching following stroke segmentation is to provide evidence in a given metric to find the best matches of given lexicon entries in a next word matching stage. All possible combinations of strokes within a permissible window are sent to a character recognizer. And the character recognizer provides corresponding confidence projected in its knowledge base. In our implementation, distance between feature vector extracted from the given combination of strokes and built-in prototypes of associated character subset, is used as a metric.

Following the stroke spacing model described in [5], it is assumed that a character has at least one prime stroke. The number of characters in a lexicon entry and the number of prime primitives in a given image are used to decide the basic size of matching window of each character. For more meaningful alignment of a sub-array of prime primitives to a character, filtering using a weighted matching window which is a function of a prime stroke period is applied to the combination of segmented strokes for a character.

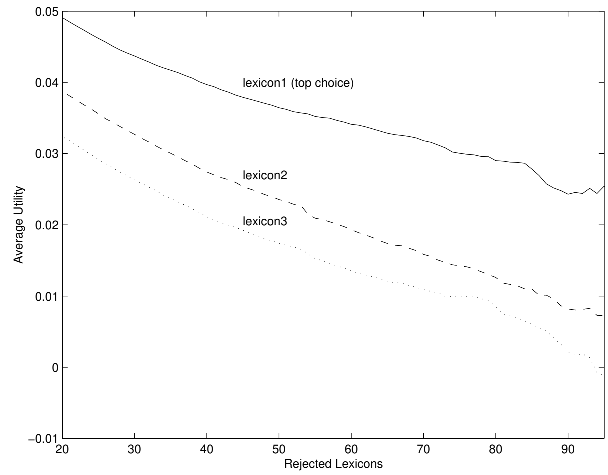


Figure 4. recognition utility

A basic permissible window size for a i_{th} character h_i of a lexicon $\mathbf{h}$ on a stroke primitive array $\mathbf{s}$ is determined as

$$
\begin{aligned}
s_{min} &= min(i+1, N_s - \eta_s(N_h - i)) \\
s_{max} &= max(\eta_s(N_h - i), N_s - (N_h - i))
\end{aligned}
$$

where s_{min} and s_{max} are the minimum and maximum of a window, N_h is the length of the lexicon $\mathbf{h}$, and η_s is the maximum allowable number of strokes for a character matching.

The actual distance measurement of a subset of symbol h in a lexicon entry $\mathbf{h}$ in a combination of all strokes between s_a and s_b is given by a product of the window function and the distance function of a character recognizer as

$$
A^{a,b}(h) = \min_{\forall c \in h} \left(\delta_h(c|s_{a\sim b})\right) \tag{15}
$$

In our experiment, $\delta_h(c|S_{a\sim b})$ generates Euclidean distance between features extracted from combined image of segments $s_a \sim s_b$ and the prototypes.

Dynamic programming technique is used to find the best fit between the possible symbol strings of a lexicon entry and the segmented stroke frame of a given image. The recognition distance found by a character recognizer is used as a metric to find word recognition distance. One of the matching paths which generates minimum distance measure throughout a possible symbol string of a lexicon entry is chosen as the best match of the lexicon entry.

5 Experiment

The proposed recognition utility is applied in a lexicon driven word recognizer for experiments. For the character

1063-6919/00 $10.00 © 2000 IEEE

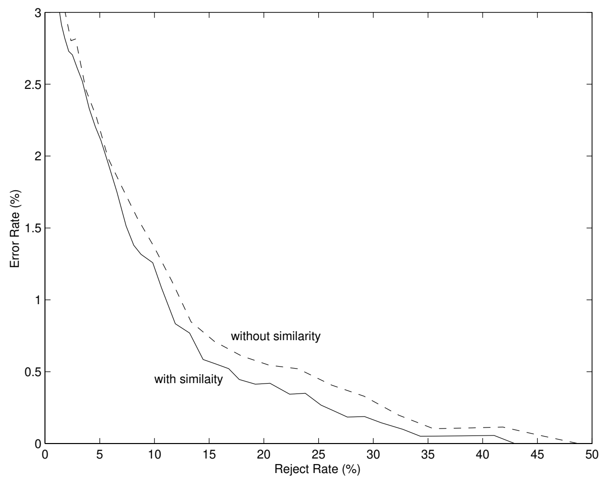


Figure 5. recognition performance

recognition engine, from setting of $N_f = 2$ and $N_g = 8$, total 36 feature measurements are extracted for a feature vector. 4 additional sub-images have been generated from quad division using a centroid of contours of a given character image. Finer level of feature vectors are obtained from the sub-images using same feature extraction method. Thus total 180 dimensional feature vector is used for measuring feature distance between given images and prototypes.

Using isolated character images from word images which are exclusive to testing word images, a maximum of 30 templates per class have been generated for each subnode by a $K - means$ clustering algorithm with a mean square error threshold of 0.1. The training character images are generated from pre-isolated character images which are manually segmented and collected from word images taken from address blocks of mail pieces in USPS main stream.

3000 word images including city names, state names, and street names collected from address block of mail pieces in USPS mail stream were used as testing images which is not used for training. The lexicons were generated as subsets which has randomly selected lexicon entries in fixed size (10 and 100) from the pool of all the possible city, state, and street names. The true word was always present in the lexicon. The word recognizer achieved 96.3% of top choice correct rate with size 10 lexicons.

Figure 4 shows dynamic characteristics of adjusted distance by liability exposure in term of lexicon complexity. The setting of the index of rejectivity, β, is 0.7, and the lexicon size is 100. If the rejected lexicon entries become larger, the distance gap between the first and second choices becomes large.

Figure 5 shows the overall effect of using lexicon complexity throughout the testing set. The lexicons of size 10 were used. The index of rejection β was set at 0.7 when liability (lexicon complexity) was applied. The recognizer using both character recognition distance and lexicon complexity (liability) has better performance compared to the recognizer using only the distance.

6 Conclusion

We have presented a method to capture lexical entropy in order to enhance recognition performance of a lexicon driven word recognizer which has a segmentation process prior to character recognition. Lexical entropy is evaluated as a parameter, *lexicon complexity*, in each lexicon entry and is reflected in the decision making as convex combination of visual feature distance metrics. Experimental results show the marginal improvement of recognition performance. Finding a generalized lexical entropy measuring method is expected as further research.

References

[1] A. W. Senior and A. J. Robinson. An Off-Line Cursive Handwriting Recognition System. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):309–321, 1998.

[2] G. Kim and V. Govindaraju. A Lexicon Driven Approach to Handwritten Word Recognition for Real-Time Applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):366–379, 1997.

[3] R. A. Wagner and M. J. Fischer. The String-to-String Correction Problem. *J. Assoc. Comput. Mach.*, 21(1):168–173, 1974.

[4] M. A. Goodrich, W. C. Stirling and R. L. Frost. A Theory of Satisficing Decisions and Control. *IEEE Transactions on System, Man and Cybernetics - Part A*, 28(6):763–779, 1998.

[5] J. Park, V. Govindaraju and S. N. Srihari. Efficient Word Segmentation Driven By Unconstrained Handwritten Phrase Recognition. In *International Conference on Document Analysis and Recognition, Bangalore, India*, pages 605–608, 1999.

[6] H. Freeman. Computer Processing of Line-Drawing Images. *Computing Surveys*, 6(1):57–97, 1974.