Computation of Normalized Edit Distance and Applications

Andrés Marzal and Enrique Vidal

***Abstract*—Given two strings X and Y over a finite alphabet, the normalized edit distance between X and Y, $d(X,Y)$ is defined as the minimum of $W(P)/L(P)$, where P is an editing path between X and Y, $W(P)$ is the sum of the weights of the elementary edit operations of P, and $L(P)$ is the number of these operations (length of P). In this paper, it is shown that in general, $d(X,Y)$ cannot be computed by first obtaining the conventional (unnormalized) edit distance between X and Y and then normalizing this value by the length of the corresponding editing path. In order to compute normalized edit distances, a new algorithm that can be implemented to work in $O(m \cdot n^2)$ time and $O(n^2)$ memory space is proposed, where m and n are the lengths of the strings under consideration, and $m \geq n$. Experiments in hand-written digit recognition are presented, revealing that the normalized edit distance consistently provides better results than both unnormalized or post-normalized classical edit distances.**

***Index Terms*—Editing, Levenshtein distance, normalized edit distance, optical character recognition, pattern recognition, speech recognition, spelling correction, string correction,**

I. Introduction

GIVEN TWO strings X and Y over a finite alphabet, an *edit distance* between X and Y can be defined as the minimun weight of transforming X into Y through a sequence of weighted *edit operations*. These operations are usually defined in terms of *insertion* of a symbol, *deletion* of a symbol, and *substitution* of one symbol for another. There are a number of well-known algorithms for computing edit distances [13], [7], [10] and/or for solving other more-or-less directly related problems [4], [10], [11], [14]. Many of these algorithms find their usefulness in error correcting, pattern recognition, and other related applications [4], [3], [10]. Nevertheless, the edit distances, as defined so far, are not very suitable for many of these applications since they lack some type of *normalization* that would appropriately rate the weight of the (edit) errors with respect to the sizes of the objects (strings) that are compared. For instance, two (edit) errors in a comparison between strings of length 3, say, are more important than three errors in a comparison of strings of length 9.

Although some straightforward heuristic normalization criteria are often more-or-less successfully applied in most practical situations, in this paper, we show that the computation of properly defined *normalized edit distances* cannot, in general, be carried out by using the algorithms that are known thus far for computing edit distances. In order to compute these normalized edit distances, a new algorithm is introduced. This algorithm is shown to work in $O(m \cdot n^2)$ time and $O(n^2)$ memory space for strings of lengths m and n, and $n \leq m$.

Manuscript received June 13, 1991; revised April 21, 1992. This work was supported by the Spanish CICYT under grant TIC-0448/89 and by a grant from the Spanish "Ministerio de Educatión y Ciencia." Recommended for acceptance by Associate Editor R. De Mori.

The authors are with the Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Valencia, Spain.

IEEE Log Number 9209987.

II. Review of Edit Distances

Let Σ be a finite alphabet and Σ^* be the set of all finite-length strings over Σ. Following a notation similar to that used in the classical paper of Wagner and Fisher [13], let $X = X_1 X_2 \ldots X_n$ be a string of Σ^*, where X_i is the ith symbol of X. We denote by $X_{i\ldots j}$ the substring of X that includes the symbols from X_i to $X_j, 1 \leq i, j \leq n$. The length of such a string is $|X_{i\ldots j}| = j - i + 1$. If $i > j, X_{i\ldots j}$ is the null string $\lambda, |\lambda| = 0$.

An *elementary edit operation* is a pair $(a,b) \neq (\lambda,\lambda)$, where both a and b are strings of lengths 0 or 1, respectively. The edit operation (a,b) is often written as $a \rightarrow b$. There are three types of elementary edit operations, namely, *insertions*, *substitutions*, and *deletions*, which take the forms $\lambda \rightarrow b, a \rightarrow b$, and $a \rightarrow \lambda$, respectively. An *edit transformation* of X into Y is a sequence S of elementary edit operations that transforms X into Y. Edit transformations are also known as "*listings*" [10]. An example of edit transformation is given in Fig. 1(c). Elementary edit operations can be weighted by an arbitrary weight function γ that assigns to each elementary operation $a \rightarrow b$ a nonnegative real number $\gamma(a \rightarrow b)$. The function γ can be extended to edit transformations $S = S_1 S_2 \ldots S_m$ by letting $\gamma(S) = \sum_{i=1}^{m} \gamma(S_i)$.

Given $X, Y \in \Sigma^*$, the edit distance between X and Y is then defined as

$$\delta(X,Y) = \min \{\gamma(S) | S \text{ is an edit transformation of } X \text{ into } Y\}. \tag{2.1}$$

A direct consequence of this definition is that edit distances fulfill the triangle inequality, regardless of whether such a property holds for the elementary weights or not. Correspondingly, $\delta(.,.)$ is a metric over Σ^* if the following conditions are imposed to $\gamma : (a \rightarrow a) = 0, \gamma(a \rightarrow b) > 0$ if $a \neq b$, and $\gamma(a \rightarrow b) = \gamma(b \rightarrow a), \forall a, b, c \in \Sigma \cup \{\lambda\}$.

Edit distances can also be defined in terms of *Traces*, where a Trace from X to $Y, T_{X,Y}$, or simply T if X and Y are

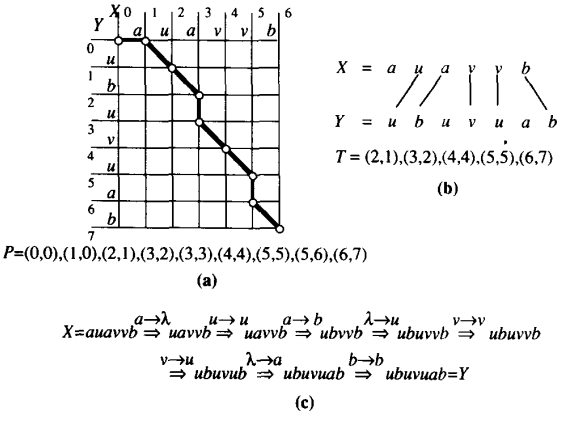

Fig. 1. Editing path (a) between strings X, Y, along with the corresponding trace (b) and edit transformation (c). Diagonal path segments in (a) correspond to substitutions, whereas horizontal and vertical segments represent deletions and insertions, respectively.

understood, is a set (sequence) of ordered pairs of integers (i, j) satisfying the following (Fig. 1(b)):

1) $1 \leq i \leq |X|; 1 \leq j \leq |Y|$.
2) for every two distinct pairs $(i, j), (i', j')$ in $T_{X,Y}$: $i < i' \Leftrightarrow j < j'$.

Weights can be assigned to Traces by means of the weighting function γ:

$$W(T_{X,Y}) = \sum_{(i,j)\in T_{X,Y}} \gamma(X_i \to Y_j) + \sum_{i\in I} \gamma(X_i \to \lambda) + \sum_{j\in J} \gamma(\lambda \to Y_j) \quad (2.2)$$

where I and J are the sets of positions of X and Y, respectively, which are not related through T.

If the triangle inequality holds for the elementary weight function γ, an important result of Wagner and Fischer [13] relates *edit distances* as defined in (2.1) with *Traces*:

$$\delta(X, Y) = \min\{W(T) | T \text{ is a trace from } X \text{ to } Y\}. \quad (2.3)$$

Trace weights can be more compactly specified through an alternative notation that follows from the concept of the *editing path*.

An editing path between X and Y, $P_{X,Y}$, or simply P if X and Y are understood, is a sequence of *points* or ordered pairs of integers $(i_k, j_k), 0 \leq k \leq m$ satisfying the following:

$$\begin{aligned} &\text{(a) } 0 \leq i_k \leq |X|; 0 \leq j_k \leq |Y|; \\ &\qquad (i_0, j_0) = (0, 0); (i_m, j_m) = (|X|, |Y|) \\ &\text{(b) } 0 \leq i_k - i_{k-1} \leq 1; 0 \leq j_k - j_{k-1} \leq 1, \forall k \geq 1 \\ &\text{(c) } i_k - i_{k-1} + j_k - j_{k-1} \geq 1. \end{aligned} \quad (2.4)$$

Fig. 1 shows an example of an editing path between two strings along with the associated trace and edit transformation.

Every pair of successive points of an editing path corresponds to an elementary edit operation so that, using the above stated convention that $i > j \Rightarrow z_{i\ldots j} = \lambda \forall z \in \sum^*$, we can associate weights to paths as follows:

$$W(P_{X,Y}) = \sum_{k=1}^{m} \gamma(X_{i_{k-1}+1} \ldots i_k \to Y_{j_{k-1}+1} \ldots j_k) \quad (2.5)$$

where $P_{X,Y} = (i_0, j_0), \ldots, (i_k, j_k), \ldots, (i_m, j_m)$.

From the above result of Wagner and Fischer (2.3) and the direct relation existing between Traces and Paths, it is easily seen that

$$\delta(X, Y) = \min\{W(P) | P \text{ is an editing path between } X \text{ and } Y\}. \quad (2.6)$$

The recursive relation due to Wagner and Fischer

$$\begin{aligned} \delta(X_{1\ldots i}, Y_{1\ldots j}) = \min\{&\delta(X_{1\ldots i-1}, Y_{1\ldots j}) + \gamma(X_i \to \lambda), \\ &\delta(X_{1\ldots i-1}, Y_{1\ldots j-1}) + \gamma(X_i \to Y_j), \\ &\delta(X_{1\ldots i}, Y_{1\ldots j-1} + \gamma(\lambda \to Y_j)\} \end{aligned} \quad (2.7)$$

leads directly, through dynamic programming, to iterative procedures for computing $\delta(X, Y)$ in $O(|X| \cdot |Y|)$ time and memory space [13]. Furthermore, if the actual editing paths or traces are not needed (only the edit distances are really required), the actual space requirements can be easily reduced to $O(\min(|X|, |Y|))$.

III. Normalized Edit Distances

Given an editing Path $P = (i_0, j_0) \ldots (i_m, j_m)$, let the length of P, $L(P)$ be defined as the number of elementary edit operations described by P, that is, $L((i_0, j_0) \ldots (i_m, j_m)) = m$. Correspondingly, the *normalized weight* of a (nonnull) path P is

$$\hat{W}(P) = \frac{W(P)}{L(P)} \quad (3.1)$$

where $W(P)$ is defined in (2.5).

The *normalized edit distance* between two strings X and Y is defined as

$$d(X, Y) = \min\{\hat{W}(P) | P \text{is an editing path between } X \text{ and } Y\} \quad (3.2)$$

and is associated with $\hat{W}(P)$.

The normalized edit distance has been defined here directly in terms of *paths* (or *traces*) rather than *edit transformations*. In fact, unless certain nontrivial conditions are imposed on the elementary edit weight function γ and/or on the definition of edit sequences, no meaningful definition of normalized edit distance seems possible in terms of edit transformations. For instance, if γ is zero for certain pairs of symbols, then for any two strings X, Y, there could be infinitely long sequences of elementary edit operations with normalized weight equal zero.

On the other hand, it should be noted that the minimization (3.2) can by no means be carried out by first minimizing $W(P)$ through (2.7) and then normalizing it by the length of the

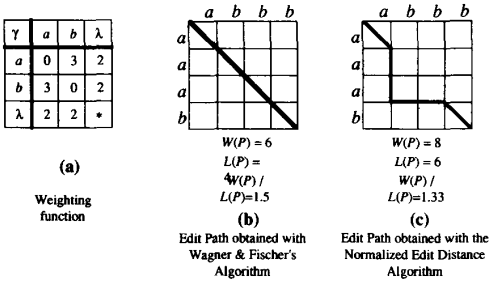

Fig. 2. Example of edit distance with post-normalization versus normalized edit distance.

obtained path, as in (3.1). The following example illustrates how such a *post-normalization* procedure produces a wrong result.

Example 3.1: Let $X = abbb, Y = aaab$, and γ be as specified in Fig. 2(a). The result of minimizing $W(P)$ through (2.7) is shown in Fig. 2(b). The weight so obtained is $W(P) = 6$, and the length of the corresponding path is $L(P) = 4$. The ratio $W(P)/L(P) = 1.5$ is greater than $8/6 \approx 1.33$, which is the actual normalized edit distance between X and Y, as defined in (3.2) (see Fig. 2(c)). □

Although metric properties directly follow from definition (2.1) for the conventional edit distance, certain difficulties appear in the case of normalization. In fact, post-normalization is clearly nontriangular, as is shown by the counterexample of Fig. 3 for the same γ function as in Example 3.1 (Fig. 2). On the other hand, although the (correct) normalized edit distance d seems more likely to fulfill the triangle inequality, the arguments of Section II are no longer applicable here to support this property and, in this case as well, counterexamples can be found. One of these counterexamples can be obtained using a γ function in which the sum of the costs of deleting and inserting a particular symbol is (much) smaller than any other elemental edit cost. For instance, if $\Sigma = \{a, b\}, \gamma(a, a) = \gamma(b, b) = 0, \gamma(a, b) = \gamma(b, a) = \gamma(a, \lambda) = \gamma(\lambda, a) = 5$ and $\gamma(b, \lambda) = \gamma(\lambda, b) = 1$, then for $X = a, Y = ab$, and $Z = b$ we have $d(X, Y) + d(Y, Z) = 1/2 + 7/3 \not\geq 3 = d(X, Z)$, Fortunately enough, practical situations are generally less contrived and, as will be discussed in Section V, triangular behavior has actually been observed in practice for the (correctly) normalized edit distance.

IV. Efficient Computation of Normalized Edit Distances

A straightforward procedure for computing $d(X, Y)$ would ask for expanding all the possible editing paths between X and Y and computing the corresponding normalized weights. Obviously, this would lead to exponential computing time. However, one may realize that from all such (a exponential number of) paths, only a very small number of *sets of paths* of different lengths is possible. This leads to an efficient algorithm for computing $d(X, Y)$.

The basic idea is to compute one minimum edit weight for each of the possible lengths of editing paths. Once all these weights are available, they can be divided by their

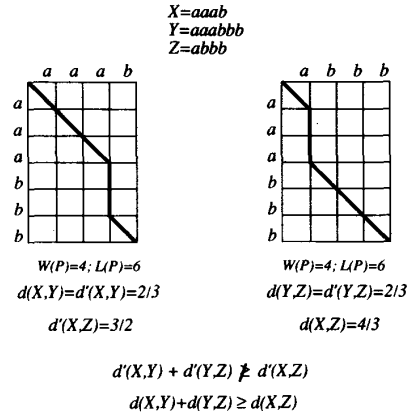

Fig. 3. Counterexample of triangle inequality satisfaction for the post-normalized edit distance (d'). The weighting function γ and strings X and Z are the same as those of Example 3.1. Although the triangle inequality fails for post-normalized edit distance d', it is fulfilled for the (correct) normalized edit distance d.

corresponding path lengths, and the minimum quotient can be given as the normalized edit distance. The following lemma establishes how many different edit path lengths are possible when two strings are compared.

Lemma 4.1: Let P be any editing path between $X_{1\ldots i}$ and $Y_{1\ldots j}$. The length of P is then bounded as

$$\max(i, j) \leq L(P) \leq i + j.$$

Proof:

$$P = (i_0, j_0) \ldots (i_k, j_k) \ldots (i_m, j_m)$$

where

$$(i_0, j_0) = (0, 0), (i_m, j_m) = (i, j), \text{ and } m = L(P).$$

From (2.4b)

$$m \geq \sum_{k=1}^{m} i_k - i_{k-1} = i_m - i_0 = i$$
$$m \geq \sum_{k=1}^{m} j_k - j_{k-1} = j_m - j_0 = j.$$

Therefore

$$m \geq \max(i, j).$$

From (2.4c)

$$m \leq \sum_{k=1}^{m} (i_k - i_{k-1} + j_k - j_{k-1}) = i_m - i_0 + j_m - j_0 = i + j.$$

□

As a consequence of Lemma 4.1, the number of different lengths of editing paths between X and Y is

$$N = \min(|X|, |Y|) + 1. \quad (4.1)$$

Definition 4.1: Let $\mathcal{P}_{i,j}$ be the set of all editing paths between $X_{1\ldots i}$ and $Y_{1\ldots j}$, and let $D(i,j,k) = \min\{W(P)|P \in \mathcal{P}_{i,j} \wedge L(P) = k\}$; with $D(i,j,k) = \infty$ if $P \in \mathcal{P}_{i,j}|L(P) = k$.

Theorem 4.1: Let $n = \max(|X|,|Y|)$, $m = |X| + |Y|$. Then

$$D(x,y) = \min_{n \le k \le m} \frac{D(|X|,|Y|,k)}{k}.$$

Proof: (From the definition of normalized edit distance (3.2), Lemma 4.1, and Definition 4.1):

Let $\mathcal{P}$ be the set of editing paths between X and Y. Then

$$\begin{aligned} d(X,Y) &= \min\left\{\frac{W(P)}{L(P)}|P \in \mathcal{P}\right\} \\ &= \min\left\{\bigcup_{n \le k \le m}\left\{\frac{W(P)}{L(P)}|P \in \mathcal{P} \wedge L(P) = k\right\}\right\} \\ &= \min_{n \le k \le m}\left\{\min\left\{\frac{W(P)}{L(P)}|P \in \mathcal{P} \wedge L(P) = k\right\}\right\} \\ &= \min_{n \le k \le m}\left\{\frac{D(|X|,|Y|,k)}{k}\right\} \end{aligned}$$

Theorem 4.2: *(Recursive Relation):*

$$\forall i,j,\ 1 \le i \le |X|,\ 1 \le j \le |Y| : \forall k : \max(i,j) \le k \le i+j$$

$$\begin{aligned} D(i,j,k) = \min\{&D(i-1,j,k-1) + \gamma(X_i \to \lambda), \\ &D(i,j-1,k-1) + \gamma(\lambda \to Y_j), \\ &D(i-1,j-1,k-1) + \gamma(X_i \to Y_j)\} \end{aligned}$$

$$D(i,j,k) = \infty \quad \forall k < \max(i,j), \forall k > i+j.$$

Proof:

$$\begin{aligned} &D(i,j,k) \\ &= \min\{W(P)| = (0,0),\ldots,(i,j) \wedge L(P) = k\} \\ &= \min\{ \\ &\quad \{W(P)|P = (0,0),\ldots,(i-1,j),(i,j) \wedge L(P) = k\} \cup \\ &\quad \{W(P)|P = (0,0),\ldots,(i,j-1),(i,j) \wedge L(P) = k\} \cup \\ &\quad \{W(P)|P = (0,0),\ldots,(i-1,j-1),(i,j) \wedge L(P) = k\} \\ &= \min(\\ &\quad \min\{W(P)|P = (0,0),\ldots,(i-1,j),(i,j) \wedge L(P) = k\}, \\ &\quad \min\{W(P)|P = (0,0),\ldots,(i,j-1),(i,j) \wedge L(P) = k\}, \\ &\quad \min\{W(P)|P = (0,0),\ldots, \\ &(i-1,j-1),(i,j) \wedge L(P) = k\}) \\ &= \min(\\ &\quad \min\{W(P)|P = (0,0),\ldots,(i-1,j) \wedge L(P) = k-1\} \\ &\quad + \gamma(X_i \to \lambda), \\ &\quad \min\{W(P)|P = (0,0),\ldots,(i,j-1) \wedge L(P) = k-1\} \\ &\quad + \gamma(\lambda \to Y_j), \\ &\quad \min\{W(P)|P = (0,0),\ldots,(i-1,j) \wedge L(P) = k-1\} \\ &\quad + \gamma(X_i \to Y_j)) \\ &= \min(D(i-1,j,k-1) + \gamma(X_i \to \lambda), \\ &\qquad D(i,j-1,k-1) + \gamma(\lambda \to Y_j), \\ &\qquad D(i-1,j-1,k-1) + \gamma(X_i \to Y_j)). \end{aligned}$$

According to Lemma 4.1, no P exists between $X_{1\ldots i}, Y_{1\ldots j}$ with $L(P) = k$ for $k < \max(i,j)$ or $k > i+j$. Therefore, from Definition 4.1, for all such k, $D(i,j,k) = \infty$. □

Theorem 4.3:

(a) $\forall i, 1 \le i \le |X|$:

$$D(i,0,i) = \sum_{l=1}^{i} \gamma(X_l \to \lambda) \text{ and } D(i,0,k) = \infty \forall k \ne i.$$

(b) $\forall j, 1 \le j \le |Y|$:

$$D(0,j,j) = \sum_{l=1}^{i} \gamma(\lambda \to Y_l), \text{and} D(0,j,k) = \infty \forall k \ne j.$$

Proof:

a) From Lemma 4.1, there exists only one editing path P between $X_{1\ldots i}$ and $Y_{1\ldots 0}$. The length of this path is $L(P) = i$, and since $Y_{1\ldots 0} = \lambda$, P is only composed of the deletions of all the symbols of $X_{1\ldots i}$.

b) Similar to a). □

Using the recursive relations of Theorem 4.2 and Theorem 4.3, $D(|X|,|Y|,k)$ can be computed through dynamic programming for all k such that $\max(|X|,|Y|) \le k \le |X| + |Y|$. From Theorem 4.1, this can lead to an algorithm for computing $d(X,Y)$ in $O(|X| \cdot |Y| \cdot \min(|X|,|Y|))$ time. A direct implementation of this algorithm for computing $d(X,Y)$ requires an array of $(|X|+1) \cdot (|Y|+1) \cdot (|X|+|Y|)$ memory locations for storing the successively computed values of D. However, from (4.1), one can take advantage of the fact that only $\min(i,j) + 1$ different lengths are possible for editing paths between $X_{1\ldots i}$ and $Y_{1\ldots j}$ to reduce the array to a size of $(|X|+1) \cdot (|Y|+1) \cdot (\min(|X|,|Y|)+1)$ by appropriately indexing the k entries of this array. The editing path associated with $d(X,Y)$ can be easily obtained from the previously computed array of values of D with no change in the order of either the time or space complexity growth functions. Further memory reduction is possible if only the value of the normalized edit distance between X and Y is required and not the corresponding editing path. In this case, only those values of D that are to be used in the next step of the main loop need be stored, yielding an algorithm with memory complexity in $O(\min(|X|,|Y|)^2)$. Two implementations corresponding to the first and last previously discussed versions of the proposed algorithm are presented in the Appendix.

With the ordering of subproblems (i,j,k) that has been adopted in these implementations, the computation is performed *on-line* with one of the given strings. Obviously, other orderings are possible, leading to different implementations. One of these orderings is the (perhaps most "natural") *multistage* ordering with the stages corresponding to the successive possible values of path lengths k. This leads to (not on-line) implementations with identical computational costs as those discussed above [6].

γ	0	1	2	3	4	5	6	7	λ
0	0.00	6.31	7.22	8.61	8.61	9.71	7.14	6.82	3.38
1	6.28	0.00	6.09	9.27	∞	8.58	8.17	8.17	3.18
2	7.14	6.20	0.00	6.77	7.39	9.34	8.24	8.24	3.22
3	∞	8.42	6.48	0.00	6.34	7.32	7.32	∞	2.67
4	9.69	9.69	7.75	6.80	0.00	7.05	7.90	9.69	3.58
5	7.65	8.57	9.26	∞	6.43	0.00	6.27	8.16	3.04
6	7.14	9.34	8.65	8.65	8.24	5.76	0.00	6.70	3.34
7	6.16	7.37	7.77	∞	∞	8.47	5.83	0.00	2.78
λ	3.69	3.77	3.67	4.18	3.52	3.66	3.88	4.18	∞

Fig. 4. Insertion, deletion, and substitution weight function that has been used in the experiments.

V. Experimental Results

In order to compare the appropriateness of the correctly normalized edit distance versus that of post-normalized or unnormalized edit distances, a practical pattern recognition problem that consists of hand-written digit recognition through edit-distance based nearest-neighbor classification has been considered.

The data consists of 500 strings (50 per digit) that represent the contours of the isolated sample digits obtained from several writers. These digits were captured through a rather standard image-acquisition procedure and chain coded into strings representing their outer contours. The alphabet of these strings is the conventional one of eight symbols ($\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7\}$), each for one of the eight possible 45° directions and lengths that define the discrete contours over a grid with a resolution of 8 pixels. Some examples of these strings are shown in Fig. 6.

The weight function γ required for the elementary edit operations was obtained through a (dynamic-programming-based) learning technique known as the error correcting grammatical inference (ECGI) [9] from a set of 15 chain-coded digits of each class. The ECGI technique gives, as a byproduct, a probability matrix for substitutions of any pair of symbols of the alphabet, as well as for insertions and deletions of any symbol. This probability matrix was properly transformed into a weight function by computing the negative logarithm of each probability value, except for $\gamma(a \rightarrow a), a \in \Sigma$, whose values have always been set to zero. The matrix that resulted from this operation is shown in Fig. 4.

This weight function was used to compute edit distances between samples in three different ways: unnormalized, post-normalized, and normalized. The first one is the classical edit distance. The second method carries out the normalization by computing the quotient between the classical unnormalized edit distance and the length of the longest edit path that yields the optimal unnormalized edit cost. Finally, the third procedure is the normalized edit distance that has been introduced in this paper.

A classification experiment based on the nearest neighbor rule was performed with each of these edit distance methods. In each case, the number of randomly chosen prototypes per class was varied from 1 to 20, and the rest of the data were used for testing. The resulting correct classification rates for the different edit distances are graphically displayed in Fig. 5, which shows a consistent superiority of the normalized distance over both unnormalized and post-normalized edit distances.

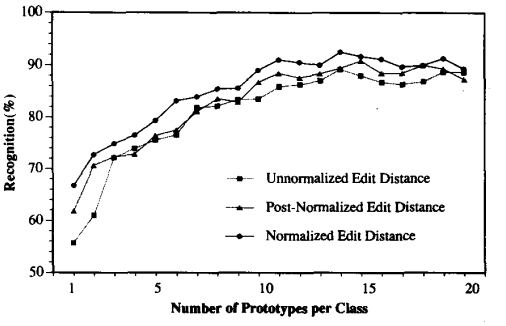

Fig. 5. Results for hand-written digits recognition experiment.

X=321254544225644446000050110006660

Y=345454442570501101006510 2

Z=34354444544444460001000005667000111 2

X'=32144454545246454544000101667222010101066660

	δ	d'	d
(X,Y)	**67.40**	1.87	1.83
(X,Z)	80.51	**1.83**	1.78
(X,X')	85.92	1.87	**1.76**

Fig. 6. Example of misclassification with both unnormalized (δ) and post-normalized edit distances (d') and correct classification with the normalized edit distance (d). A sample of the digit "7" on the top is closer to a prototype of "2" when δ is used, whereas prototype "9" is the nearest neighbor for d'. Only the normalized edit distance (d) leads to proper classification in this case.

An example of correct classification with normalized edit distance and misclassification with both unnormalized and post-normalized edit distances is shown in Fig. 6. When a string is compared with others, the unnormalized edit distance (δ) tends to yield, in general, smaller values for short strings. The post-normalized edit distance (d') attenuates this effect, but it tends to yield smaller values for comparisons between similar-length strings. Only the normalized edit distance (d) is really independent of the *length of the comparison*.

Apart from these recognition experiments, an additional set of experiments was carried out in order to empirically establish the extent to which the triangle inequality is fulfilled by the different edit distances. To this end, for each possible triplet in our 500 strings data set, the *triangle inequality looseness* [12]

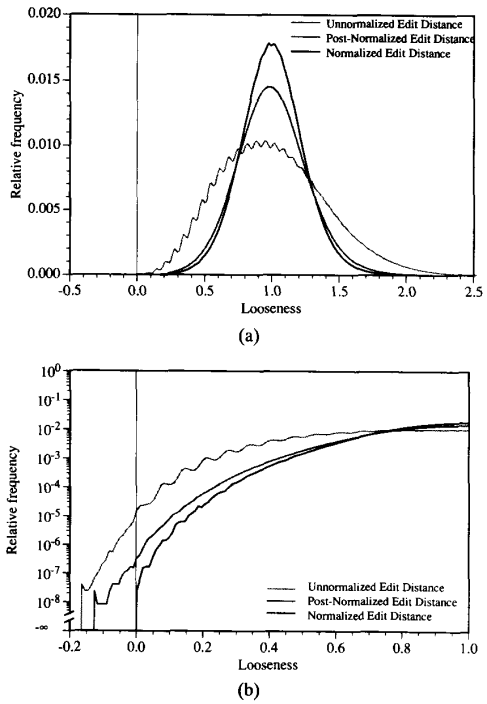

Fig. 7. (a) Histograms of the *triangle inequality looseness* for the unnormalized, post-normalized, and normalized edit distance; (b) close-up of the same histograms around *looseness*=0 in logarithmic scale. Only the normalized edit distance strictly fulfills the triangle inequality. The *looseness* values are normalized by their corresponding *average distance* between strings.

$H(X,Y,Z) = \Delta(X,Y) + \Delta(Y,Z) - \Delta(X,Z)$ was computed for each of the three edit distances ($\Delta \in \{$ unnormalized, post-normalized, normalized$\}$). These values were normalized by the corresponding average distance between strings (computed for all pairs of strings). The resulting histograms appear in Fig. 7, showing that only the normalized edit distance strictly satisfies the triangle inequality for all the triplets.

VI. Concluding Remarks

In this paper, a correct procedure for computing normalized edit distances has been presented with a linear increase in computational complexity with respect to the classical unnormalized edit distance procedure. This correctly normalized edit distance has been shown to clearly outperform both the unnormalized and the (suboptimal) post-normalized edit distances in a pattern recognition problem using the nearest neighbor classification rule.

Normalization appears to be an important aspect to take into account in many pattern recognition problems approached through edit distances. Even a suboptimal or incorrect incorporation of normalization in the classical edit distance algorithm, such as the post-normalization technique, has proven useful to improve the results with respect to those obtained with the unnormalized edit distance.

There are many other (suboptimal) normalization techniques that have been proposed and are often used in practical situations. For instance, in automatic speech recognition, normalization by (the sum of) the lengths of the compared strings is quite popular in *dynamic time warping* [1], [8], [12], which is a dynamic programming procedure that is closely related to string editing [10]. Another suboptimal normalization technique that has been proposed in the speech recognition field consists of minimizing at each point of the computational lattice the quotient of the current distance by the current path length [2], [5].

Since all of these suboptimal techniques are computationally cheaper than the optimal one proposed here, experimental work is required in order to determine to what extent these techniques could be appropriate in each specific pattern recognition task and whether a correct normalization does in fact

```
Algorithm Normalized Edit Distance
input     X,Y ∈ Σ+ ;
output    d : real ;
function  γ : (Σ∪{λ}) × (Σ∪{λ}) → real ;
var       D : array [0...|X|,0...|Y|,0...|X|+|Y|+1] of real ;
            // the two extra values of the third index are intended for simplifying
               control and enhancing readability //
          i, j, k : integer ;
begin
  D[0,0,0] := 0 ; D[0,0,1] := ∞ ;
  for j := 1 to |Y| do
    D[0,j,j-1]:= ∞ ; D[0,j,j] :=D[0,j-1,j-1]+γ(λ → Yj); D[0,j,j+1]:= ∞;     //Theorem 4.3b//
  endfor
  for i := 1 to |X| do
    D[i,0,i-1]:= ∞; D[i,0,i]:=D[i-1,0,i-1]+γ(Xi → λ); D[i,0,i+1]:= ∞;       //Theorem 4.3a//
    for j := 1 to |Y| do
      D[i,j,max(i,j)-1] := ∞                                                //Theorem 4.2//
      for k := max(i,j) to i+j do                                           //Lemma 4.1//
        D[i,j,k] := min( D[i-1,j,k-1] + γ(Xi → λ ),                         //Theorem 4.2//
                         D[i,j-1,k-1] + γ(λ → Yj),
                         D[i-1,j-1,k-1] + γ(Xi → Yj) )
      endfor
      D[i,j,i+j+1] := ∞                                                     //Theorem 4.2//
    endfor
  endfor
  d := ∞
  for k := |X| to |X|+|Y| do d := min (d, D[|X|,|Y|,k] / k) endfor          //Theorem 4.1//
end.
```

Fig. 8. Algorithm 1.

```
Algorithm Normalized Edit Distance
input     X,Y ∈ Σ+ ;   // |Y|≤|X| //
output    d : real ;
const     Previous = 0 ; Current = 1 ;
function  γ : (Σ∪{λ}) × (Σ∪{λ}) → real ;
var       D : array [Previous...Current,0...|Y|,0...|Y|+2] of real ;
            // the two extra values of the third index are intended for simplifying
               control and enhancing readability //
          i, j, k, P, C, ofs, ofsi, ofsj, ofsij: integer ;
begin
  P := Previous; C := Current;
  D[P,0,0] := ∞ ; D[P,0,1] := 0 ; D[P,0,2] := ∞ ;
  for j := 1 to |Y| do
    D[P,j,0] := ∞ ; D[P,j,1] := D[P,j-1,1] + γ(λ → Yj) ; D[P,j,2] := ∞
  endfor
  for i := 1 to |X| do   // 'C' represents 'i' and 'P' represents 'i-1' //
    D[C,0,0] := ∞ ; D[C,0,1] := D[P,0,1] + γ(Xi → λ) ; D[C,0,2] := ∞
    for j := 1 to |Y| do
      ofs := max(i,j)-1
      ofsi := max(i-1,j)-1; ofsj := max(i,j-1)-1; ofsij := max(i-1,j-1)-1
      D[C,j,0] := ∞ ;     // D[C,j,max(i,j)-1-ofs] //
      for k := max(i,j) to i+j do
        D[C,j,k-ofs]:=min( D[P,j,k-1-ofsi] + γ(Xi → λ ),
                           D[C,j-1,k-1-ofsj] + γ(λ → Yj),
                           D[P,j-1,k-1-ofsij] + γ(Xi→Yj) )
      endfor
      D[C,j,i+j+1-ofs] := ∞
    endfor
    (P,C) := (C,P)
  endfor
  d := ∞
  for k := |X| to |X|+|Y| do d := min (d, D[P,|Y|,k-|X|+1] / k) endfor
end.
```

Fig. 9. Algorithm 2.

lead to greater recognition accuracy. Some of these experiments are currently in progress in our laboratories. Finally, future investigation should also address the problem of fast computation of the normalized edit distance.

APPENDIX
ALGORITHMS

This Appendix shows two implementations corresponding to the first and last versions of the proposed algorithm. Fig. 8 shows Algorithm 1: Space complexity $O(|X|\cdot|Y|\cdot(|X|+|Y|))$. Fig. 9 shows Algorithm 2: Space complexity $O(|Y|^2)$.

ACKNOWLEDGMENT

The image data was provided by J. M. Valiente and G. Andreu from DISCA of the Universidad Politénica de Valencia. The probabilities for elementary edit operations were supplied by H. Rulot from CIUV of the Universidad de Valencia. The authors gratefully acknowledge all these contributions to this work.

REFERENCES

[1] F. Casacuberta and E. Vidal, *Reconocimiento Automático del Habla*. Barcelona: Marcombo, 1987.
[2] J. Di Martino, "Dynamic time warping algorithms for isolated and connected word recognition," in *New Systems and Architectures for Automatic Speech Recognition and Synthesis* (R. De Mori and Y. Suen, Eds.). Berlin: Springer Verlag, 1985.
[3] K. S. Fu, *Syntactic Pattern Recognition and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1982.
[4] P. A. V. Hall and G. R. Dowling, "Approximate string matching," *ACM Comput. Surveys*, vol. 12, pp. 381–402, Dec. 1980.
[5] Y. Kitazume, E. Ohira, and T. Endo, "LSI implementation of a pattern matching algorithm for speech recognition," *IEEE Trans. Acoustics Speech Signal Processing*, vol. 33, no. 1, pp. 1–5, Feb. 1985.
[6] A. Marzal and E. Vidal, "On the computation of normalized edit distances revisited," Tech. Rep. DSIC-II/15/1991, Depto. de Sistemas Informáticos y Computación, Univ. Politécnica de Valencia.
[7] W. J. Masek and M. S. Patterson, "A faster algorithm computing string edit distances," *J. Comput. Syst. Sci.*, vol. 20, pp. 18–31, Feb. 1980.
[8] L. Rabiner and L. Levinson, "Isolated and connected word recognition—Theory and selected applications," *IEEE Trans. Commun.*, vol. C-29, no. 5, pp. 621–659, 1981.
[9] H. Rulot and E. Vidal, "Modelling (Sub)string-length-based constraints through a grammatical inference method," in *Pattern Recognition: Theory and Applications* (Devijver and Kittler, Eds.). Berlin: Springer Verlag, 1987, pp. 451–459.
[10] D. Sankoff and J. B. Kruskal, *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Reading, MA: Addison-Wesley, 1983.
[11] P. H. Sellers, "The theory and computation of evolutionary distances: Pattern recognition," *J. Algorithms*, vol. 1, pp. 359–373, 1980.
[12] E. Vidal, F. Casacuberta, J. M. Benedi, M. J. Lloret, and H. Rulot, "On the verification of triangle inequality by dynamic time-warping dissimilarity measures," *Speech Commun.*, vol. 7, pp. 67–69, 1988.
[13] R. A. Wagner and M. J. Fischer, "The string-to-string correction problem," *J. Assoc. Comput. Machinery*, vol. 21, no. 1, pp. 168–173, Jan. 1974.
[14] Y. P. Yang and T. Pavlidis, "Optimal correspondence of string subsequences," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. 12, no. 11, pp. 1080–1087, Nov. 1990.

Andrés Marzal was born in Valencia, Spain, on June 7, 1966. He received the Licenciado degree in computer science from the Facultad de Informática of the Universidad Politécnica de Valencia in 1990. He is currently working towards the Ph.D. degree in computer science at the Departamento de Sistemas Informáticos y Computación of the Universidad Politécnica de Valencia.

His current research interests are in pattern recognition and algorithms for automatic speech recognition.

Enrique Vidal received the Licenciado en Ciencias Físicas degree in 1978 and the Doctor en Ciencias Físicas degree in 1985, both from the Universidad de Valencia.

From 1972 to 1978, he was with several different companies working in electronics and computer engineering. In 1978, he joined the Computer Center of the Universidad de Valencia, where he served as a systems analyst, and in 1981, he joined the Departamento de Electrónica e Informática of the same university as an honorary collaborator. After that, he coordinated, in both centers, a research group in the field of automatic speech recognition. In 1986, he left the Universidad de Valencia and joined the Departamento de Sistemas Informáticos y Computación of the Universidad Politécnica de Valencia, where, up until now, he has served as a Profesor Titular of the Facultad de Informática. His current fields of interest include statistical and syntactic pattern recognition and their applications to automatic speech recognition, where it is especially concerned with grammatical inference and, in general, with automatic learning methodologies.

Dr. Vidal is a member of the International Association for Artificial Intelligence (AEPIA). He also serves as a member of the governing board of the Spanish Society for Pattern Recognition and Image Analysis (SERFAI), which is an affiliate society of IAPR. He is co-author of the book *Reconocimiento Automático del Habla*, which was awarded the *Mundo Electrónico* Prize for the best technical book in 1985.