INFORMATION SCIENCES 23, 123–142(1981)

An Effective Algorithm for String Correction Using Generalized Edit Distances— I. Description of the Algorithm and Its Optimality*

R. L. KASHYAP
and
B. J. OOMMEN

School of Electrical Engineering, Purdue University, West Lafayette, Indiana 47907

Communicated by John M. Richardson

ABSTRACT

This paper deals with the problem of estimating a transmitted string X_s from the corresponding received string Y, which is a noisy version of X_s. We assume that Y contains any number of substitution, insertion, and deletion errors, and that no two consecutive symbols of X_s were deleted in transmission. We have shown that for channels which cause independent errors, and whose error probabilities exceed those of noisy strings studied in the literature [12], at least 99.5% of the *erroneous strings* will not contain two consecutive deletion errors. The best estimate X^+ of X_s is defined as that element of H which minimizes the generalized Levenshtein distance $D(X/Y)$ between X and Y. Using dynamic programming principles, an algorithm is presented which yields X^+ without computing individually the distances between every word of H and Y. Though this algorithm requires more memory, it can be shown that it is, in general, computationally less complex than all other existing algorithms which perform the same task.

I. INTRODUCTION

One of the problems encountered in text recognition is that of correcting misspelled words. Let Y be a misspelled (noisy) string obtained from a word X_s, an element of a finite dictionary H. Various algorithms have been proposed to obtain an appropriate estimate of X_s based on Y.

Many researchers [7, 9, 10, 13, 21] have defined X^+, the best estimate of X_s, as that $X \in H$ which minimizes the Levenshtein distance $D(X/Y)$ between X and Y. The standard technique (ST) of computing X^+ requires the separate evaluation of the edit distance between Y and every element $X \in H$. However,

*Partially supported by the National Science Foundation under Grant No. NSF 78-18271.

52 Vanderbilt Ave., New York, NY 10017 0020-0255/81/02123+20$02.50

the ST does not utilize the information it has obtained in the process of evaluating any one $D(X_j/Y)$, to compute any other $D(X_k/Y)$. Suppose X_j and X_k have the same prefix $X^{(P)}=a_1a_2\ldots a_p$. Then the ST would compute the distance $D(a_1\ldots a_r/Y)$ for both X_j and X_k and would thus unnecessarily repeat the same comparisons and minimizations for all $r=1,\cdots,P$. Thus, the ST usually has many redundant computations.

In this paper we present a new technique, called Algorithm I, to compute $X^+\in H$ which minimizes $D(X/Y)$ for a given Y. In contrast to the ST, in Algorithm I we do *not* individually evaluate $D(X_i/Y)$ for every $X_i\in H$. By treating the dictionary as one integral unit and by using dynamic programming principles, we compute the distances $D(X/Y)$ for all $X\in H$ *simultaneously*. In Algorithm I we take the maximum advantage of the information contained in the prefixes of the words of the dictionary. In other words, we make the most of the fact that the recognizer for a finite dictionary is a finite state machine (FSM) which has an inherent tree structure. The computational advantage of Algorithm I over other existing algorithms [7, 9, 10, 13, 20, 21] is considered in the companion paper [18].

The heart of the paper is the recursive computation of the distance $D(X/Y)$. To render the computation recursive, we introduce an auxiliary measure $D_1(X/Y)$, termed the pseudodistance between X and Y, from which $D(X/Y)$ can be computed using one additional symbol comparison. The distance measure $D_1(X/Y)$ has some desirable properties. Let $X^{(i)}$ be the prefix of X of length i, and let $Y^{(j)}$ be the prefix of Y of length j. The pseudodistance measure between $X^{(i)}$ and $Y^{(j)}$ can be computed using *only a fixed finite number* of pseudodistances between the prefixes of $X^{(i)}$ and $Y^{(j-1)}$ respectively.

In Sec. II, we define the distance $D(X/Y)$ and obtain an explicit expression for it. In the next section we prove its recursive properties. Using dynamic programming procedures [4], we proceed to show how X^+ can be obtained recursively. The fact that the structure of this FSM that accepts H can be represented as a tree, is used in the companion paper [18] to study the computational complexity of Algorithm I. A comparison of Algorithm I with other existing algorithms is also found in [18].

II. FUNDAMENTALS AND DEFINITIONS

II.1. NOTATION

An alphabet A is a finite set of symbols, and A^* is the set of strings over A. $Z\in A^*$ is a string of length $|Z|$. A dictionary H is a predefined set of J words:

$$H=\left\{X_i=\prod_{k=1}^{N_i} x_{ik} \mid i=1,\ldots,J;\ x_{ik}\in A\right\},$$

where Π is a concatenation relation.

N_m is the length of the longest word in H. The null string μ is defined by (i) $\alpha\mu=\mu\alpha=\alpha$ and (ii) $\mu\mu=\mu$, where $\alpha\in A^*$. We distinguish λ, the null symbol, from μ, the null string. Let $\hat{A}=A\cup\{\lambda\}$. $\hat{A}$ is called the *appended alphabet*. Uppercase letters will be used to represent strings of symbols, and lowercase letters to represent elements of the alphabet under consideration.

Let $Z\in A^*$ be a string of length R, $R\geqslant 0$. Then Z_p, the left derivative of Z of order one (or simply the left derivative of Z) is defined as the prefix of Z of length $R-1$, if $R\geqslant 1$, and as μ if $R=0$. Z_g, the left derivative of Z_p, is referred to as Z's left derivative of order two.

II.2. TRANSMISSION ERRORS

Suppose a string $X_s\in H$ is the input to a noisy channel, and Y is the output noisy string. We assume that there are three types of errors in Y (let $\alpha,\beta\in A^*$ and $a,b\in A$, where $a\neq b$):

(i) Y has a substitution error if $X_s=\alpha a\beta$ and $Y=\alpha b\beta$.
(ii) Y has a deletion error if $X_s=\alpha a\beta$ and $Y=\alpha\beta$.
(iii) Y has an insertion error if $X_s=\alpha\beta$ and $Y=\alpha a\beta$.

The following are the assumptions made regarding the properties of the noisy channel through which X_s is transmitted.

ASSUMPTION i. The noise that corrupts any x_i in X_s is independent of the noise that corrupts any other symbol x_j in X_s.

ASSUMPTION ii. The channel has the property that it does not delete two consecutive symbols of X_s. In Sec. III.6 we relax the assumption and require that no P consecutive symbols of X_s be deleted in the process of transmission, where $2\leqslant P<N_s$ (N_s is the length of X_s).

We shall justify Assumption ii in Sec. III.5.

II.3. DISTANCES BETWEEN STRINGS

Suppose we edit a received string $W\in A^*$ and transform it to another string $Z\in A^*$ by either replacing some of its symbols with other symbols in A, or inserting some new symbols in it, or deleting some of its symbols. We can then define a distance $D(Z/W)$ between Z and W, based on the intersymbol elementary edit distance measure $d(\cdot/\cdot)$ obeying the inequalities of Okuda et al. [13]. The meaning of these intersymbol edit distances is given below.

(i) $d(a/b)$ is the edit distance associated with replacing a b in W with an a to transform it to Z, $0\leqslant d(a/b)\leqslant\infty$. By definition $d(a/a)=0$.

(ii) $d(\lambda/b)$ is the edit distance associated with deleting a b in W, where $0<d(\lambda/b)<\infty$.

(iii) $d(b/\lambda)$ is the edit distance associated with inserting a b in W, where $0<d(b/\lambda)<\infty$.

The distance $D(Z/W)$ is defined as the minimum sum of the distances associated with the edit operations done on W to transform it to Z. We now obtain an explicit expression for it. Suppose that both Z and W are of the same length M. If z_i and w_i (the ith symbols of Z and W respectively) are such that $d(z_i/w_i)<\infty$ for all $i=1,\ldots,M$, then the definition of $D(Z/W)$ yields the following expression:

$$D(Z/W)=\sum_{i=1}^{M} d(z_i/w_i).$$

Suppose Z and W are not of the same length. We can create two strings Z' and W', *both of the same length* L', by inserting λ's between the symbols of Z and W respectively, and can set up a one to one correspondence between the symbols z_i' and w_i', $i=1, 2,\ldots, L'$, the individual symbols of Z' and W' respectively. Thus with each such pair (Z', W') we can associate a unique set of edit operations which is that of transforming each w_i' to the corresponding z_i', provided both are not simultaneously λ. Further, with each pair we can associate a sum of distances given by

$$\sum_{i=1}^{L'} d(z_i'/w_i').$$

The minimum of the sum of the edit distances over all the possible pairs (Z', W') obeying Assumption ii is the distance $D(Z/W)$. A precise expression for it is given by

$$D(Z/W)=\min_{(Z', W')\in T''}\left[\sum_{i=1}^{L'} d(z_i'/w_i')\right], \qquad Z, W\in A^*, \tag{2.1}$$

where $T''=\{(Z', W')|\, Z', W' \text{ satisfying (a)–(c)}\}$:

(a) Z' and W' are strings obtained by inserting λ's in Z and W respectively, with no $z_i'=w_i'=\lambda$.

(b) The symbols in W' corresponding to any two consecutive symbols of Z are not both λ, due to Assumption ii.

(c) $\max[R, K]\leqslant|Z'|=|W'|=L'\leqslant R+K-1$, $R=|Z|$, $K=|W|$.

The above expression explicitly defines the generalized Levenshtein distance that has been used by previous researchers [4, 5, 7, 9, 13, 21, 22].

III. THE COMPUTATION OF $D(Z/W)$

Our primary intention is to develop a recursive procedure to compute $D(Z/W)$ which involves *only* a fixed finite number of the prefixes of Z and the left derivative of W. To this end we introduce a distance measure, $D_1(Z/W)$, referred to as the pseudodistance between Z and W. The measure $D_1(Z/W)$ has the desirable properties that it can be computed recursively and that $D(Z/W)$ can be obtained from it using only one additional symbol comparison. The pseudodistance $D_1(Z/W)$ between $Z \in A^*$ and a received $W \in A^*$ is defined as the minimum sum of the individual edit distances associated with the edit operations needed to transform a received W to Z *given that the last symbol of* Z *was not inserted during editing.* It can be evaluated by minimizing the sum of the edit distances over all pairs in T' of (2.1) which do not require the *edit insertion* of Z_R, the last symbol of Z. Thus,

$$D_1(Z/W) = \min_{(Z', W') \in T_1} \left[\sum_{i=1}^{L'} d(z_i'/w_i') \right], \tag{3.1}$$

where $T_1 = \{(Z', W') | Z', W'$ satisfying (a)–(c) below (2.1) and (d) below$\}$:

(d) If $z_i' = z_R$ then $w_i' \neq \lambda$

The relationship between $D_1(Z/W)$ and $D(Z/W)$ is given by the following theorem, proved in the Appendix:

THEOREM I. *Let* $Z = \prod_{i=1}^{R} z_i$ *and* $W = \prod_{i=1}^{K} w_i$. *Let* Z_p *be the left derivative of* Z. *Then the following distance equality is true*:

$$D(Z/W) = \min\left[D_1(Z/W), \{ D_1(Z_p/W) + d(z_R/\lambda) \} \right]. \tag{3.2}$$

Let $Y^{(K)}$ be the prefix of Y of length K. We now derive the recursive properties of the pseudodistance between an arbitrary string $Z \in A^*$ and $Y^{(K)}$. The cases when $|Z| = 0$, $|Z| = 1$, and $|Z| \geq 2$ have been considered separately.

Using the definition of μ, we have the following expression:

$$D_1(\mu/Y^{(K+1)}) = D_1(\mu/Y^{(K)}) + d(\lambda/y_{K+1}). \tag{3.3}$$

Suppose we want to compute the pseudodistance between a single symbol b and $Y^{(K+1)}$. By the definition of the pseudodistance b could not have been an

inserted symbol. Using Lemma I and simple dynamic programming arguments,

$$D_1(b/Y^{(K+1)}) = \min\Big[\big\{D_1(b/Y^{(K)}) + d(\lambda/y_{K+1})\big\},$$

$$\big\{D_1(\mu/Y^{(K)}) + d(b/y_{K+1})\big\}\Big]. \tag{3.4}$$

The expression for the pseudodistance between Z and $Y^{(K+1)}$ when $|Z| \geq 2$ is given by the following theorem proved in the Appendix:

THEOREM II. *Let Z_1bc be an element of A^*, $b, c \in A$, and $|Z_1| \geq 0$. Then $D_1(Z_1bc/Y^{(K+1)})$ can be evaluated by the following equation:*

$$D_1(Z_1bc/Y^{(K+1)}) = \min\Big[\big\{D_1(Z_1bc/Y^{(K)}) + d(\lambda/y_{K+1})\big\},$$

$$\big\{D_1(Z_1b/Y^{(K)}) + d(c/y_{K+1})\big\},$$

$$\big\{D_1(Z_1/Y^{(K)}) + d(b/\lambda) + d(c/y_{K+1})\big\}\Big]. \tag{3.5}$$

In the above expression the number of terms included to obtain the distance between Z_1bc and $Y^{(K+1)}$ is merely three. Such a simplified expression is a consequence of (i) performing the recursion using the pseudodistance measure, and (ii) utilizing the fact that the channel obeys Assumption ii. It is exactly here that the distance computation presented here is superior to the one presented in [20].

III.1. PROCEDURE FOR OBTAINING X^+

Let V be the set of all the prefixes of H. Then $\alpha \in V$ if and only if α is a concatenation of the first r symbols of some word in H. ($r \leq N_m$, the length of the longest word in H.) Symbolically,

$$V = \{\alpha \mid \alpha\beta \in H;\ \alpha, \beta \in A^*\}.$$

We define $R^{(K)}$ as the set of all elements of V into which $Y^{(K)}$ can be transformed with finite pseudodistance. For all $K = 1, \ldots, M$ where $M = |Y|$,

$$R^{(K)} = \big\{Z_1 \mid Z_1 \in V, D_1(Z_1/Y^{(K)}) < \infty\big\}.$$

Similarly, we define $S^{(K)}$ as

$$S^{(K)}=\{(Z_1,u)|\,D_1(Z_1/Y^{(K)})=u<\infty\}.$$

The reason why we have defined $R^{(K)}$ and $S^{(K)}$ in terms of the pseudodistances $D_1(Z_1/Y^{(K)})$ is that they are recursively computable. Let us assume that we have the sets $R^{(K)}$ and $S^{(K)}$ obtained after editing the string $Y^{(K)}$. Then, using Assumption ii, we can show that $R^{(K+1)}$ can contain only terms of the form Z_1, Z_2c, Z_1bc, where $Z_1, Z_2 \in R^{(K)}$, $b, c \in A$, and $d(c/y_{K+1})<\infty$. Hence, the set $R^{(K+1)}$ can be computed from a knowledge of just $R^{(K)}$, V, and the last incoming symbol y_{K+1}. This result is stated below and proved in the Appendix:

THEOREM III. *Let $Y^{(K)}$ be the prefix of Y of length K. If $R^{(K)}$ is the set of all $Z \in V$ for which $D_1(Z/Y^{(K)})<\infty$, then, using Assumption ii, $R^{(K+1)}$ can be written as the union of three mutually exclusive sets*:

$$R^{(K+1)}=R^{(K)}\cup R'\cup R'', \tag{3.6}$$

where,

$$R'=\{Z_2c|\,Z_2c\in V,\ Z_2c\notin R^{(K)},\ Z_2\in R^{(K)},\ d(c/y_{K+1})<\infty\},$$

$$R''=\{Z_1bc|\,Z_1bc\in V,\ Z_1b\notin R^{(K)},\ Z_1bc\notin R^{(K)},\ Z_1\in R^{(K)},\ d(c/y_{K+1})<\infty\},$$

$Z_1, Z_2 \in V$ *and* $b, c \in A$.

Combining the results of Theorem III and (3.3)–(3.5), one can see that $S^{(K)}$ is also recursively computable. Using these results, we present Algorithm I that processes Y and yields as its output X^+.

Initially we define $R^{(0)}=\{\mu\}$ and $S^{(0)}=\{(\mu,0)\}$. From the results obtained above, we know that if the sets $R^{(K)}$ and $S^{(K)}$ are known, and the incoming symbol y_{K+1} is known, then the sets $R^{(K+1)}$ and $S^{(K+1)}$ can be computed directly. Thus, if $|Y|=M$, $Y^{(M)}$ can be obtained recursively. After the whole string Y has been processed, we create two new sets R^+ and S^+ defined below:

$$R^+=\{X|\,X\in R^{(M)}\cap H\}\cup\{X|\,X\notin R^{(M)},\ X\in H,\ X_p\in R^{(M)};$$

$$X_p \text{ the left derivative of } X\},$$

$$S^+=\{(X,u)|\,X\in R^+,\ u=D(X/Y)\}.$$

The set R^+ has the property that every element in it is strictly an element of H and it satisfies the inequality $D(X/Y)<\infty$. Similarly, in the set S^+ we retain only the pairs (X, u), where $X\in H$ and u is the *distance* $D(X/Y)<\infty$. By observing S^+, the best estimate $X^+\in H$ which minimizes $D(X/Y)$ can be obtained. The procedure verbally stated above is given algorithmically below:

ALGORITHM I.

1. Initialize $R^{(0)}=\{\mu\}$, $S^{(0)}=\{(\mu,0)\}$.
2. For $K=0,\ldots,M-1$ do
   (a) Compute $R^{(K+1)}$ from $R^{(K)}$ and y_{K+1}, using (3.6).
   (b) Compute $S^{(K+1)}$ from $S^{(K)}$ and $R^{(K+1)}$ as

$$S^{(K+1)}=\left\{\left(Z, D_1\left(Z/Y^{(K+1)}\right)\right)\middle| Z\in R^{(K+1)}\right\},$$

   where $D_1(Z/Y^{(K+1)})$ is obtained using (3.3), (3.4), and (3.5) if $|Z|=0$, $|Z|=1$, and $|Z|\geq 2$ respectively.
3. Compute R^+ and S^+ from $R^{(M)}$ and $S^{(M)}$ as:
   (a) $R^+=\{X|X\in R^{(M)}\cap H\}\cup\{X|X\in H, X\notin R^{(M)}, X_p\in R^{(M)}\}$
   (b) $S^+=\{(X,u)|X\in R^+\}$, in which u is the distance $D(X/Y)$ obtained using (3.2).
4. Choose $X^+\in R^+$ as the string which Y represents which minimizes $D(X/Y)$.

III.2. EXAMPLE

To illustrate Algorithm I an example is given below for the dictionary H where

$$H=\{format, or\},$$

$$A=\{a,f,g,m,o,r,t\}.$$

Let the elementary edit distances be given by $d(b/b)=0$ for all $b\in A$, $d(f/g)=3.4$, $d(b/\lambda)=2.3$ for all $b\in A$, $d(\lambda/b)=2.3$ for all $b\in A$. For simplicity we have assumed that all the other intersymbol distances associated with substitution are infinite.

Suppose $Y=gormt$. Without much comment, we briefly give the sets and complete the example.

1. $R^{(0)}=\{\mu\}$. $S^{(0)}=\{(\mu,0)\}$.
2. $y_1=$'g'. Thus,

$$R^{(1)}=\{f,\mu\},$$

$$S^{(1)}=\{(f,d(f/g)),(\mu,d(\lambda/g))\}=\{(f,3.4),(\lambda,2.3)\}.$$

3. $y_2 = 'o'$. Hence,

$$R^{(2)} = \{fo, o, f, \mu\},$$

$$S^{(2)} = \{(fo,3.4),(o,2.3),(f, d(f/g)+d(\lambda/o)),(\mu,4.6)\}$$

$$= \{(fo,3.4),(o,2.3),(f,5.7),(\mu,4.6)\}.$$

4. $y_3 = 'r'$.

$$R^{(3)} = \{for, or, fo, o, f, \mu\}$$

$$S^{(3)} = \{(for,3.4),(or,2.3),(fo,5.7),(o,4.6),(f,8.0),(\mu,6.9)\}$$

5. $y_4 = 'm'$.

$$R^{(4)} = \{form, for, fo, f, or, o, \mu\},$$

$$S^{(4)} = \{(form,3.4),(for,5.7),(fo,8.0),(f,10.3),(or,4.6),(o,6.9),$$

$$(\mu,9.2)\}.$$

6. $y_5 = 't'$.

$$R^{(5)} = \{format, form, fo, f, or, o, \mu\},$$

$$S^{(5)} = \{(format,5.7),(form,5.7),(for,8.0),(fo,10.3),(f,12.6),$$

$$(or,6.9),(o,9.2),(\mu,11.5)\}.$$

7. $R^+ = H$, since no insertions can be made to create any new words which are not in $R^{(5)}$ and yet are in H. Thus by observation $D(format/gormt) = 5.7$ and $D(or/gormt) = 6.9$, whence we decide that $X^+ = format$.

REMARK. Trimming the size of $R^{(K)}$ and $S^{(K)}$ by rejecting all states $Z \in R^{(K)}$ for which $D_1(Z/Y^{(K)}) \geq D_0$ (a threshold) considerably reduces the computation time. The threshold D_0 can be either a constant or a function of K.

III.3. A SIMPLIFIED VERSION OF ALGORITHM I

In this section we show that the recognizer for a finite dictionary is a finite state machine (FSM) which has an inherent tree structure. We then make use of

the fact that Algorithm I merely manipulates the nodes of this tree and present a simplified form of Algorithm I which is more easily programmable.

Let $H^{(P)}$ be the set of all the prefixes of H of length less than or equal to P. By definition, V is the set of all the prefixes of H. Thus,

$$H^{(P)} = \{\alpha \mid |\alpha| \leq P, \alpha\beta \in H, \alpha, \beta \in A^*\}, \tag{3.7}$$

implying that

$$V = H^{(N_m)}. \tag{3.8}$$

The FSM that accepts only words in H is given by the quintuple $(A, V, T, f, H^{(0)})$, where:

(i) A is the finite alphabet.

(ii) V is the set of states defined above.

(iii) T is the set of final states $= H$.

(iv) f is the transition function which defines the next state, if the present state and the input symbol $b \in A$ are given. f is thus a map from $V \times A$ to V. $f(\alpha, b) = \alpha b$ if and only if α is the present state, b is the input symbol, and both α and αb are in V.

(v) $H^{(0)}$ is the starting state μ.

We now explicitly define the tree structure of the FSM that accepts only words in H. Since H is a finite set, this FSM must have no cycles. Being a completely connected graph, the transition map of the FSM can thus be represented by a tree having the following features:

(i) The nodes of the tree correspond to the elements of V.

(ii) If Z is a node of the tree, then by virtue of the transition function of the FSM, Z_p will be the parent node of Z, and Z_g will be the grandparent of Z.

(iii) The root of the tree will be the node corresponding to μ.

(iv) The leaves of the tree will all be words in H (but the converse is not true).

In future, we do not distinguish between the FSM and its corresponding tree. Neither do we distinguish between the nodes of the tree and their labels.

With this tree structure as the basis we can now view Theorems II and III in a different way. Theorem III states that if a certain node Z of the tree is in $R^{(K)}$, then the nodes that are in $R^{(K+1)}$ caused by Z are merely all the children and the grandchildren of Z. Similarly, Theorem II states that if the pseudodistance between a certain node Z and $Y^{(K+1)}$ has to be computed, it can be done with merely a knowledge of the pseudodistances between each of Z, the parent of Z, the grandparent of Z, and the string $Y^{(K)}$ respectively. This justifies the simplified version of Algorithm I below.

Let α be an element of V whose left derivatives of order one and two are α_p and α_g respectively. Further, for any $X \in H$, let X_p be its parent and x_f its last symbol. Then Algorithm I can be seen to be equivalent to the procedure given below.

SIMPLIFIED VERSION OF ALGORITHM I.

Input: (1) The dictionary H in terms of the sets $H^{(i)}$ for all $i \leq N_m$ (the length of the longest word in H) and the tree structure of the FSM that accepts H.
(2) The garbled string Y.

Output: The string $X^+ \in H$ which minimizes $D(X/Y)$.

Method:

$D_1(\mu/Y^{(0)})=0$, where $Y^{(0)}=\mu$.

for $K=1$ to M do
  if $(2K<N_m)$ $S=2K$
  else $S=N_m$
  for every $\alpha \in H^{(S)}$ do
    if ($D_1(\alpha/Y^{(K-1)})$ or $D_1(\alpha_p/Y^{(K-1)})$ or $D_1(\alpha_g/Y^{(K-1)})<\infty$) then
      $q1=D_1(\alpha/Y^{(K-1)})+d(\lambda/y_K)$
      $q2=D_1(\alpha_p/Y^{(K-1)})+d(a_f/y_K)$
      $q3=D_1(\alpha_g/Y^{(K-1)})+d(a_{pf}/\lambda)+d(a_f/y_K)$
      $D_1(\alpha/Y^{(K)})=\text{Min}[q1,q2,q3]$
  end
end for every $X \in H$ do
    $D(X/Y)=\text{Min}[D_1(X/Y), D_1(X_p/Y)+d(x_f/\lambda)]$
end $X^+=\text{Arg}\ \underset{X}{\text{Min}}[D(X/Y)]$

end

The "if" statement in the second "for" loop tests whether it is possible for α to be in $R^{(K)}$. Independent of $d(a_f/y_k)$, $D(\alpha/Y^{(K)})$ is computed. If $D(\alpha/Y^{(K)}) = \infty$, then α will not be in $R^{(K)}$ and hence its contribution in the subsequent computations will be neglected. When all three, α, α_p, $\alpha_g \notin R^{(K-1)}$, α can never be in $R^{(K)}$ and hence $D(\alpha/Y^{(K)})$ is not evaluated. The last "for" loop considers the possibility of x_f being deleted in transmission. One can verify that, by virtue of the results of Theorems II and III, the above algorithm is equivalent to Algorithm I.

III.5. JUSTIFICATION OF ASSUMPTION II

In the formulation of Algorithm I we have assumed that in the process of transmission no two consecutive symbols of X_s were deleted. In this subsection we justify this assumption.

Let p_c be the probability of correctly transmitting any one symbol $a \in A$. Let p_w be the probability that the symbol is erroneously transformed into some other symbol $b \in A$. Let p_d be the probability that the symbol a is deleted. Since any transmitted symbol can either be correctly transmitted or erroneously transformed or deleted,

$$p_c + p_w + p_d = 1.$$

Let us suppose that a string X of length N is transmitted. Let E_1 be the event that at least one symbol was erroneously transformed or deleted. Let E_2 be the event that at least two consecutive deletions took place. Using Assumption i,

$$P(\text{all the symbols are correctly transmitted}) = p_c^N.$$

Hence,

$$P(E_1) = 1 - p_c^N.$$

To evaluate $P(E_2)$, we observe that there are 2^N distinct possibilities of either deleting or not deleting the N transmitted symbols. By enumerating the 2^N possibilities and evaluating the probabilities of the individual elementary events that favor E_2, $P(E_2)$ can be computed. Since E_2 is a subevent of E_1, $P(E_2/E_1)$ is the ratio $P(E_2)/P(E_1)$.

The probability $P(E_2/E_1)$ has been evaluated for various values of p_c, p_w, and p_d for various values of N. For example, if $p_c = 0.955$, $p_w = 0.040$, $p_d = 0.005$, and $N = 8$, we obtain that $P(E_2/E_1) = 0.00057$. This implies that 99.943% *of the erroneous strings* will *not* contain two consecutive deletions. Consider the case when $p_c = 0.94$, $p_w = 0.05$, $p_d = 0.01$, and $N = 6$. The error probabilities in this case are higher than the average error rates considered by Neuhoff [12], and the length of the words is about the average length of the 1021 most common English words [23]. Even in this case, we obtain that 99.840% *of the erroneous strings* will *not* contain two consecutive deletions. Hence, if the noisy channel obeys Assumption ii, the algorithm presented here can be used to correct all the possible noisy strings. Even if Assumption ii is not exactly satisfied, by the results derived above, we believe that the algorithm can, on the average, be used to correct at least 99.5% of the erroneous strings.

III.6. GENERALIZATION OF ALGORITHM I

We shall now generalize Algorithm I for the case when the channel does not delete P consecutive symbols of X_s. The fact that we have assumed that the

noisy channel obeys Assumption ii implies that the only term in $R^{(K+1)}$ obtained as a result of inserting a symbol after $\alpha \in R^{(K)}$ is αbc [where $d(c/y_{K+1}) < \infty$]. Suppose we relax the assumption to require that in the process of transmission no P consecutive symbols be deleted. In such a case if $\alpha \in R^{(K)}$, terms in $R^{(K+1)}$ caused by insertions are of the form $\alpha b_1 b_2 \ldots b_{P-1} c$ [where every $b_i \in \hat{A}$, $d(c/y_{K+1}) < \infty$]. The upper bound for P is $N_s - 1$, where N_s is the length of X_s.

If Assumption ii were true, then to evaluate $D_1(Z/Y^{(K+1)})$ the only distances we would need to know are $D_1(Z/Y^{(K)})$, $D_1(Z_p/Y^{(K)})$, and $D_1(Z_g/Y^{(K)})$, where Z_p and Z_g are the left derivatives of Z of order one and two respectively. Relaxing the algorithm to permit up to $P-1$ consecutive deletions will imply that we have to consider all the left derivatives of order P. This will mean that the distance expression (3.5) will contain $P+1$ terms over which the minimum must be taken. We now present Algorithm I-G, a generalization of Algorithm I, which yields as its output $X^+ \in H$ which minimizes the distance $D(X/Y)$ subject to the constraint that no P consecutive symbols of X_s were deleted in transmission. The proof that the sets $R^{(K)}$ and $S^{(K)}$ are correctly computed follows from a direct generalization of the results of Theorems II and III.

ALGORITHM I-G

1. Initialize $R^{(0)} = \{\mu\}$, $S^{(0)} = \{\mu, 0)\}$.
2. For $K = 0, \ldots, M-1$ do
   (a) For every $\alpha \in R^{(K)}$, add β to $R^{(K+1)}$, where $\beta \in V$ has α as its prefix and satisfies $|\beta| \leqslant |\alpha| + P + 1$.
   (b) Compute $S^{(K+1)}$ from $S^{(K)}$ and $R^{(K+1)}$ as

$$S^{(K+1)} = \left\{ \left(Z, D_1\left(Z/Y^{(K+1)} \right) \right) \middle| Z \in R^{(K+1)} \right\},$$

   where $D_1(Z/Y^{(K+1)})$ is obtained using (3.3), (3.4) if $|Z| = 0$ and $|Z| = 1$ respectively. If $|Z| \geqslant 2$, let one possible representation of Z be $\alpha_1 \alpha_2 c$, where $|\alpha_2| < P$. Then $D_1(Z/Y^{(K+1)})$ is the minimum of $D_1(Z/Y^{(K)})$ and P other terms each of which corresponds to the transformation of y_{K+1} to c, the insertion of α_2, and the editing of $Y^{(K)}$ to α_1.
3. Compute R^+ and S^+ from $R^{(M)}$ and $S^{(M)}$ as:
   (a) $R^+ = \{X | X \in R^{(M)} \cap H\} \cup \{X | X \in H, X \notin R^{(M)}, X_p \in R^{(M)}\}$,
   (b) $S^+ = \{(X, u) | X \in R^+\}$, in which u is the distance $D(X/Y)$ obtained using (3.2).
4. Choose $X^+ \in R^+$ as the string which Y represents which minimizes $D(X/Y)$.

IV. CONCLUSIONS

In this paper we have presented an algorithm, referred to as Algorithm I, which has as its input a misspelled string Y which represents some unknown word X_s of a finite dictionary H. The best estimate X^+ of X_s is defined as that string X in H which minimizes the generalized Levenshtein distance $D(X/Y)$ between X and Y. Algorithm I yields X^+ without individually evaluating every $D(X_i/Y)$.

By studying Algorithm I using the tree structure of the finite state machine that accepts H, we can show [18] that though it requires more memory, it minimizes the number of computations required to obtain X^+. This is because it utilizes the maximum information obtained during the process of evaluating any one $D(X_i/Y)$ to compute any other $D(X_j/Y)$. In the second part of this paper [18] it is shown that, in general, Algorithm I is computationally less complex than both the standard technique (ST) of obtaining X^+ and the algorithm presented in [20]. Its superiority has been demonstrated for various dictionaries, including the dictionary consisting of the 1021 most common English words of length greater than unity.

APPENDIX

A. PROOF OF THEOREM I

Consider the expression for $D(Z/W)$ given in (2.2). We have required that in every pair (Z', W') in T'', (a) no $z_i' = w_i' = \lambda$ and (b) the symbols in W' corresponding to any two consecutive symbols in Z be not both λ. This condition is implicitly assumed in all the sets defined in this proof. To simplify notation, let $L_1 = \max[R, K]$ and $L_2 = R + K - 1$.

We partition T'' into two mutually exclusive subsets T_a and T_b as $T'' = T_a \cup T_b$, where

$$T_a = \{(Z', W') \mid |Z'| = |W'| = L';\ L_1 \leq L' \leq L_2;\ \text{and if } z_i' = z_R, \text{ then } w_i' \neq \lambda\},$$

$$T_b = \{(Z', W') \mid |Z'| = |W'| = L';\ L_1 \leq L' \leq L_2;\ \text{and if } z_i' = z_R, \text{ then } w_i' = \lambda\},$$

Consequently,

$$D(Z/W) = \min\left[\min_{T_a}\left\{\sum_{i=1}^{L'} d(z_i'/w_i')\right\}, \min_{T_b}\left\{\sum_{i=1}^{L'} d(z_i'/w_i')\right\}\right]. \quad \text{(A.1)}$$

But T_a is identically equal to the set T_1 of (3.1). Hence the first term in (A.1) is $D_1(Z/W)$. Since in every pair (Z', W') in T_b, the symbol z_R is an inserted symbol, in view of Assumption ii, z_{R-1} cannot be an inserted symbol. Let T_b' be the set

$$\{(Z_p', W') \mid |Z_p'| = |W'| = L';\ L_1' \leq L' \leq L_2';\ \text{and if } z_q' = z_{R-1}, \text{ then } w_q' \neq \lambda\},$$

where $L_1' = \max[R-1, K]$ and $L_2' = (R-1)+K-1$. Then

$$\min_{T_b}\left[\sum_{i=1}^{L'} d(z_i'/w_i')\right] = \min_{T_b'}\left[\sum_{i=1}^{L'} d(z_i'/w_i')\right] + d(z_R/\lambda). \tag{A.2}$$

The first term on the right hand side of (A.2) is exactly $D_1(Z_p/W)$. Rewriting (A.1) using the simplified pseudodistance expressions for the individual terms, we obtain the result.

B. *PROOF OF THEOREM II*

The theorem is trivially true if $D_1(Z_1 bc/Y^{(K)}) = \infty$. We shall only consider the case when it is finite.

For the sake of convenience, let $Y_1 = Y^{(K)} = \prod_{i=1}^{K} y_i$, and let $Y_2 = Y^{(K+1)} = \prod_{i=1}^{K+1} y_i$. If $|Z_1| = R$, we write $Z_1 = \prod_{i=1}^{R} z_{1i}$. Since $|Z_1| = R$, from (3.2) we express the pseudodistance $D_1(Z_1/Y^{(K)})$ in terms of the individual edit distances as

$$\begin{aligned} D_1(Z_1/Y_1) &= \min_{(Z_1', Y_1') \in T_4}\left[\sum_{i=1}^{L'} d(z_{1i}'/y_{1i}')\right] \\ &= \min_{(Z_1', Y_1') \in T_4}\left[S_{Z_1'Y_1'}\right], \end{aligned} \tag{B.1}$$

where $T_4 = \{(Z_1', Y_1') \mid Z_1', Y_1' \text{ satisfying (a)–(c)}\}$:

(a) Z_1' and Y_1' are strings obtained by inserting λ's in Z_1 and Y_1 respectively, with no $z_{1i}' = y_{1i}' = \lambda$.

(b) If $z_{1p}' = z_R$ then $y_{1p}' \neq \lambda$.

(c) $\max[R, K] \leq |Z_1'| = |Y_1'| = L' \leq R+K-1$.

Conditions (a) and (b) regarding the position of λ will hereafter be taken for granted.

Let $Z_2 = Z_1 bc$. Then since $|Z_2| = R+2$, using (3.1),

$$D_1(Z_2/Y_2) = \min_{(Z_2', Y_2') \in T_5} \left[\sum_{i=1}^{L'} d(z_{2i}'/y_{2i}') \right]$$

$$= \min_{(Z_2', Y_2') \in T_5} \left[S_{Z_2', Y_2'} \right], \tag{B.2}$$

where $T_5 = \{(Z_2', Y_2') | Z_2', Y_2'$ satisfying (a)–(c)$\}$:

(a) Z_2' and Y_2' are strings obtained by inserting λ's in Z_2 and Y_2 respectively, with no $z_{2i}' = y_{2i}' = \lambda$.
(b) If $z_{2p}' = z_R$ then $y_{2p}' \neq \lambda$.
(c) $\max[R+2, K+1] \leqslant |Z_2'| = |Y_2'| = L' \leqslant R+K+2$.

Noting that c and y_{K+1} are the last non-λ symbols of Z_2' and Y_2', the set of all possible pairs (Z_2', Y_2') can now be partitioned into five mutually exclusive and exhaustive subsets {A}–{E} (in all the subsets, the index n_1 is arbitrary):

{A} contains the pairs in which c is to the left of y_{K+1}:

$$Z_2' = (Z_1 bc)'\lambda,$$

$$Y_2' = (Y_1')\, y_{K+1}. \tag{B.3}$$

{B} contains the pairs in which c and y_{K+1} are in identical positions. It can further be partitioned into two mutually exclusive and exhaustive subsets {B1} and {B2}. {B1} contains those pairs of {B} in which the last non-λ symbol of Y_1' lies to the left of b in Z_2':

$$Z_2' = Z_1' bc,$$

$$Y_2' = Y_2' \lambda\, y_{K+1}. \tag{B.4}$$

{B2} contains those elements of {B} in which at least one non-λ symbol of Y_1' is either identically placed as, or is to the right of, b in Z_2':

$$Z_2' = (Z_1' b \lambda^{n_1}) c,$$

$$Y_2' = (Y_1')\, y_{K+1}. \tag{B.5}$$

{C} contains all pairs in which y_{K+1} lies between b and c:

$$Z_2' = (Z_1' b \lambda^{n_1}) \lambda c,$$
$$Y_2' = (Y_1') y_{K+1} \lambda. \tag{B.6}$$

{D} contains the pairs in which b and y_{K+1} are identically placed:

$$Z_2' = Z_1' bc,$$
$$Y_2' = Y_1' y_{K+1} \lambda. \tag{B.7}$$

{E} contains the pairs in which y_{K+1} is to the left of b:

$$Z_2' = (Z_1') bc,$$
$$Y_2' = (Y_1' y_{K+1}) \lambda \lambda. \tag{B.8}$$

No element of either {C}, {D}, or {E} will be admissible in T_5, since every element in these sets requires the insertion of c, the last symbol of $Z_1 bc$. Every element in {E} requires that both b and c be inserted, and this further violates Assumption ii.

Thus the pseudodistance $D_1(Z_2/Y^{(K+1)})$ can be written as

$$D_1(Z_2/Y^{(K+1)}) = \min_{(Z_2', Y_2') \in T_5} \left[S_{Z_2' Y_2'} \right]$$
$$= \min \left[\min_{\{A\}} \left[S_{Z_2' Y_2'} \right], \min_{\{B1\}} \left[S_{Z_2' Y_2'} \right], \min_{\{B2\}} \left[S_{Z_2', Y_2'} \right] \right]. \tag{B.9}$$

From (B.3),

$$\min_{\{A\}} \left[S_{Z_2' Y_2'} \right] = D_1(Z_1 bc/Y_1) + d(\lambda/y_{K+1}). \tag{B.10}$$

From (B.4),

$$\min_{\{B1\}} \left[S_{Z_2' Y_2'} \right] = D_1(Z_1/Y_1) + d(b/\lambda) + d(c/y_{K+1}). \tag{B.11}$$

From (B.5),

$$\min_{\{B2\}} \left[S_{Z_1' Y_2'} \right] = D_1(Z_1 b/Y_1) + d(c/y_{K+1}). \tag{B.12}$$

Rewriting (B.9) using (B.10)–(B.12) proves the theorem.

C. PROOF OF THEOREM III

We recall the definition of $R^{(K+1)}$ as

$$R^{(K+1)} = \{ Z | D_1(Z/Y^{(K+1)}) < \infty \}.$$

We partition $R^{(K+1)}$ into two disjoint sets as

$$R^{(K+1)} = R_0 \cup R_1, \tag{C.1}$$

where

$$R_0 = \{ Z | D_1(Z/Y^{(K+1)}) < \infty, D_1(Z/Y^{(K)}) < \infty \}$$

and

$$R_1 = \{ Z | D_1(Z/Y^{(K+1)}) < \infty, D_1(Z/Y^{(K)}) = \infty \}.$$

We further partition R_1 into two mutually exclusive subsets R_2 and R_3:

$$R_2 = \{ Z | D_1(Z/Y^{(K+1)}) < \infty, D_1(Z/Y^{(K)}) = \infty$$

$$\text{and } Z = Z_2 c, \text{ where } Z_2 \in R^{(K)} \},$$

$$R_3 = \{ Z | D_1(Z/Y^{(K+1)}) < \infty, D_1(Z/Y^{(K)}) = \infty$$

$$\text{and } Z = Z_2 c, \text{ where } Z_2 \notin R^{(K)} \}.$$

The set R_3 can be equivalently written as

$$R_3 = \{ Z_1 bc | D_1(Z_1 bc/Y^{(K+1)}) < \infty, D_1(Z_1 bc/Y^{(K)}) = \infty,$$

$$D_1(Z_1 b/Y^{(K)}) = \infty \}.$$

The expression $D_1(Z/Y^{(K)}) < \infty$ implies that $D_1(Z/Y^{(K+1)}) < \infty$. Hence the set R_0 is identically equal to $R^{(K)}$.

Consider the set R_2. Since $Z_2c \in R^{(K+1)}$, the pseudodistance $D_1(Z_2c/Y^{(K+1)})$ is finite. Further, since $D_1(Z/Y^{(K)})$ is infinite, and since c, the last symbol of Z, must have been a substituted symbol, it must have been substituted for by y_{K+1}. Consequently, R_2 can be equivalently written as:

$$R_2 = \{ Z_2c \mid D_1(Z_2c/Y^{(K+1)}) < \infty, D_1(Z_2c/Y^{(K)}) = \infty,$$

$$D_1(Z_2/Y^{(K)}) < \infty, d(c/y_{K+1}) < \infty \}, \tag{C.2}$$

which is identical to R' of (3.5).

Consider the set R_3. Since $D_1(Z_1bc/Y^{(K+1)}) < \infty$, and since both $D_1(Z_1bc/Y^{(K)})$ and $D_1(Z_1b/Y^{(K)})$ are infinite, c must have been a substituted symbol, substituted for y_{K+1}. If b were a substituted symbol, then Z_1b would have been an element of $R^{(K)}$. Hence b must have been an inserted symbol. Since b is an inserted symbol, by virtue of Assumption ii, the last symbol of Z_1 must not be an inserted symbol. Thus, since (i) c is substituted for y_{K+1}, and (ii) b is an inserted symbol, and (iii) $d(Z_1bc/Y^{(K+1)}) < \infty$ we conclude that due to (C.3), $D_1(Z_1/Y^{(K)})$ is finite, and

$$D_1(Z_1bc/Y^{(K+1)}) \leq D_1(Z_1/Y^{(K)}) + d(b/\lambda) + d(c/y_{K+1}). \tag{C.3}$$

Hence Z_1 is an element of $R^{(K)}$. Thus the set R_3 can be rewritten as

$$R_3 = \{ Z_1bc \mid D_1(Z_1bc/Y^{(K+1)}) < \infty, D_1(Z_1/Y^{(K)}) < \infty,$$

$$D_1(Z_1b/Y^{(K)}) = \infty, D_1(Z_1bc/Y^{(K)}) = \infty, \text{ and } d(c/y_{K+1}) < \infty \}, \tag{C.4}$$

which is the set R'' of (3.6).

Combining (C.1)–(C.4), the theorem is proved.

REFERENCES

1. A. V. Aho, D. S. Hirschberg, and J. R. Ullmann, Bounds on the complexity of the longest common sub-sequence problem, *J. Assoc. Comput. Mach.* 23(1):1–12 (Jan. 1976).
2. A. V. Aho and J. D. Ullman, *The Theory of Parsing, Translation and Compiling*, Prentice-Hall, 1972.
3. L. W. Fung and K. S. Fu, Stochastic syntactic decoding for pattern classification, *IEEE Trans. Computers* C-24(6):662–667 (June 1975).

4. R. L. Kashyap, Syntactic decision rules for recognition of spoken words and phrases using a stochastic automaton, *IEEE Trans. Pat. Anal. and Mach. Intel.* PAMI-1(2):154–163 (April 1979).
5. W. J. Masek and M. S. Paterson, A faster algorithm computing string edit distances, *J. Comput. System Sci.* 20:18–31 (1980).
6. D. E. Knuth, *The Art of Computer Programming*, Vol. 1, Addison-Wesley, Reading, Mass., 1973, pp. 473–479.
7. A. Levenshtein, Binary codes capable of correcting deletions, insertions and reversals, *Soviet Physics Dokl.* 10(8):707–710 (Feb. 1966).
8. C. R. Litecky and G. B. Davis, A study of errors, error proneness, and error diagnosis in COBOL, *Comm. ACM*, (1):33–37 (Jan. 1976).
9. S. Y. Lu and K. S. Fu, A sentence-to-sentence clustering procedure for pattern analysis, *Proceedings of the IEEE Computer Society, 1st International Computer Software and Applications Conference*, 1977, pp. 492–498.
10. R. Lawrence and R. A. Wagner, An extension of the string to string correction problem, *J. Assoc. Comput. Mach.* 22:177–183 (1975).
11. H. L. Morgan, Spelling corrections in systems programs, *Comm. ACM* **13**(2):90–94 (1970).
12. D. L. Neuhoff, The Viterbi algorithm as an aid in text recognition, *IEEE Trans. Information Theory* IT-21(2):222–226 (1975).
13. T. Okuda, E. Tanaka, and T. Kasai, A method of correction of garbled words based on the Levenshtein metric, *IEEE Trans. Computers* C-25:172–177 (Feb. 1976).
14. E. M. Reisman and A. R. Hanson, A contextual post processing system for error correction using binary *n*-grams, *IEEE Trans. Computers* C-23:480–493 (May 1974).
15. R. S. Scowen, On detecting misspelt identifiers in FORTRAN, *Software—Practice and Experience* 7:536 (1977).
16. M. G. Thomason, Errors in regular languages, *IEEE Trans. Computers* 23:597–602 (1974).
17. R. L. Bahl and F. Jelinek, Decoding for channels with insertions, deletions, and substitutions with application to speech recognitions, *IEEE Trans. Information Theory* IT-21:404–411 (1975).
18. R. L. Kashyap and B. J. Oommen, An effective algorithm for string correction using generalized edit distances—II. Computational complexity of the algorithm and some applications *Information Sci.*, to appear.
19. J. D. Ullman, A binary *N*-gram technique for automatic correction of substitution, deletion, insertion and reversal errors in words, *Comput. J.* 20:141–147 (1977).
20. R. A. Wagner, Order-*n* correction for regular languages, *Comm. ACM*, 17:265–268 (1974).
21. R. A. Wagner and M. J. Fisher, The string to string correction problem, *J. Assoc. Comput. Mach.* 21:168–173 (1974).
22. C. K. Wong and A. K. Chandra, "Bounds for the String Editing Problem," *J. Assoc. Comput. Mach.* 23:13–16 (Jan. 1976).
23. G. Dewey, *Relative Frequency of English Speech Sounds*, Harvard U. P., 1923.

Received July 1980